AUGMENTED TRANSITION NETWORKS AS PSYCHOLOGICAL MODELS OF SENTENCE COMPREHENSION

Ronald M. Kaplan
Language Research Foundation
and
Harvard University
Cambridge, Massachusetts U.S.A.

Abstract: This paper describes the operation of an augmented recursive transition network parser and demonstrates the natural way in which perceptual strategies, based on the results of psycholinguistic experimentation, can be represented in the transition network grammatical notation. Several illustrative networks are given, and it is argued that such grammars are empirically justified and conceptually productive models of the psychological processes of sentence comprehension.

I. Introduction

During the past year a major research effort has been conducted to explore and refine the properties of an augmented recursive transition network parser (26, 27) and to develop a large-scale English grammar for the system.[1] Although our primary goal has been to construct a powerful and practical natural language processor for artificial intelligence and information retrieval applications,[2] we have also investigated the correspondence between the sentence processing characteristics of the parser and those of human speakers, as revealed by psychological experimentation, observation, and intuition. We have found that the grammatical formalism of the transition network is a convenient and natural notational system for fabricating psychological models of syntactic analysis. In the present paper we describe some of the psychologically appealing properties of the parser and illustrate how psycholinguistic experimental results can be mapped into simple transition network models. We suggest that building and testing such models can lead to a better understanding of linguistic performance.

It should be clear from the outset that we are not proposing a transition network model as a complete and sufficient representation of all aspects of language behavior. Rather, transition network models aim only at simulating the syntactic analysis component of performance: given an input string written in standard orthography, they attempt to discover the syntactic relationships holding between constituents. We ignore the myriad problems of phonetic decoding and segmentation and semantic and cognitive interpretation, as well as all the psycholinguistic and motivational complexities of speech production. It is in this limited sense that we refer to transition network grammars as sentence comprehension or perceptual models. Of course, we expect that more complete formalizations of language behavior will incorporate such independently developed syntactic analysis models.

In section II of this paper we sketch the linguistic and psycholinguistic background of our research. Section III describes the organization and operation of the transition network parser and depicts the grammatical notation, and section IV shows the representation in this notation of perceptual strategies induced from psycholinguistic data. In section V we discuss the fruitfulness of this modeling approach, Indicating some conceptual Issues that are clarified and some empirical predictions that arise from transition network formulations.

II. Transformational Grammar and Psycholinguistlcs

The process by which a native speaker comprehends and produces meaningful sentences in his language is extremely complex and, with our present body of psycholinguistic theory and data, is understood only slightly. This shortcoming of psycholinguistics exists despite the fact that advances in linguistic theory over the last decade have provided a number of crucial Insights into the formal structure of language and linguistic performance. To place augmented recursive transition network grammars in the context of previous research, we briefly survey some relevant results of linguistics and psycholinguistics.

A transformational grammar for a given language L formally defines the notion sentence of L by describing a mechanical procedure for enumerating all and only the well-formed sentences of L. With each sentence it also associates a structural description which provides a formal account of the native speaker's competence, the linguistic knowledge which underlies his ability to make Judgments about the basic grammatical relations (e.g., subject, predicate, object) and about such sentential properties as relative grammaticality, ambiguity, and synonymy. At present there is no clear agreement among lingulsts about the detailed features required for an adequate grammar, but certain principles of grammar organization are almost universally accepted: the structural description furnished for a sentence by the grammar must consist of (at least) two levels of syntactic representation (P-markers) — a deep structure and a surface structure — together with a specification of an ordered sequence of transformations which maps the deep structure of a sentence into appropriate surface structures.

Transformational theorists maintain that their formal model is not intended to give an accurate account of the psychological processes involved when a human being uses language, either speaking or comprehending. Any correlations observed between actual behavior characteristics and transformational grammars are accidental, signifying merely the fact that psychological and linguistic data are both obtained from the same class of native speakers (but Chomsky (6) weakens this assertion somewhat when he argues that acquisition data might have a bearing on the evaluation metric selected for grammars). Linguists have been very careful to distinguish the speaker's <u>competence</u>, which transformational grammars attempt to model, from his <u>performance</u>, the manner in which he utilizes his knowledge in processing sentences (6, 10, 19). Thus a transformational grammar might be allowed to generate sentences which are virtually impossible for a speaker to deal with. Most current grammars will generate [5a], assigning it the same subject-verb-object relations as are apparent in [5b]:

[5] a. The man the girl the cat the dog bit scratched loved ate ice cream.

b. The dog bit the cat that scratched the girl who loved the man who ate ice cream.

Very few native speakers would intuit that [5a] is grammatical, yet to prevent its generation, either the grammar must be greatly complicated or other sentences which native speakers do accept must be marked ungrammatical. Linguists resolve this dilemma and preserve the simplicity and generality of their grammars by claiming that native English speakers do have the basic knowledge to process [5a], which is therefore grammatical; speakers have trouble with it because their perceptual mechanisms do not provide the memory space and/or computational routines required to process it. A transformational grammar is a formal specification of the speaker's competence and has nothing to say about psychological functioning.

Despite these disclaimers, psycholinguists have been intrigued by transformational theory because it provides the most intricate and compelling explication to date of a large number of basic linguistic intuitions. Many experiments have been conducted to test the hypothesis that transformational operations will have direct, observable reflexes in psychological processing; Fodor and Garrett (8) and Bever (1) present useful reviews of this literature. A major concern of these studies has been to determine whether the <u>perceptual</u> complexity of sentences (the difficulty of comprehending and responding to them) is directly correlated with <u>derivational</u> complexity (e.g., the number of transformations required to generate them). Fodor and Garrett (8) examine this "derivational theory of complexity" in detail and conclude that the available psycholinguistic data do not offer much support for it and that the connection, if there is one, between transformational grammar and perception is not very direct at all.

Although psycholinguists have virtually abandoned their attempts to find perceptual reflexes of specific grammatical features, several studies have been successful in corroborating the psychological reality of the deep structure-surface structure distinction. MacKay and Bever (15) found that subjects respond differently to deep structure and surface structure ambiguities; Wanner (25) showed that the number of deep structure S-nodes underlying a sentence has a direct influence on the ease of prompted recall from long-term memory; and Bever (1) has reinterpreted the results of the click experiments (7) as demonstrating that deep-structure S-nodes affect the surface segmentation of a stimulus sentence. These experiments suggest that an adequate model of sentence comprehension must incorporate some mechanism for recovering a deep-structure-like representation of a given stimulus word string. This representation should explicitly denote at least such basic grammatical relationships as actor, verb, and object. More extensive empirical work should indicate whether deep structure must be even more abstract than this.

There are several other requirements for adequacy that we may impose on potential models of sentence comprehension, based on some common observations about our sentence processing abilities:

(a) A perceptual model must process strings in essentially temporal or linear order, for this is the order in which sentences are encountered in conversation and reading.

(b) It must process strings and provide appropriate analyses in an amount of time proportional to that required by human speakers. For example, since perceptual difficulty does not rapidly increase as the length of the sentence increases, the amount of time required by the model should be at most a slowly increasing function of sentence length.

(c) The model should discover anomalies and ambiguities where real speakers discover them, and for ambiguous sentences the model should return analyses in the same order as speakers do.

Whereas there are many well-known recognition procedures for programming languages and other relatively simple artificial languages, only a few algorithms have been proposed which aim at "transformational" recognition, that is, which attempt to develop appropriate deep structures from natural language surface strings. Some of these algorithms (17, 20, 28) incorporate more or less directly a linguistically motivated transformational grammar, in light of the empirical shortcomings of the derivational theory of complexity, it is not surprising that these proposals are inadequate perceptual models. A deep structure recovery strategy suggested by Kuno (14) operates independently of a transformational grammar and offers more psychological relevance, but it too has formal limitations (13). A procedure and grammatical notation recently described by Kaplan (11), based on an algorithm by Kay (12), appear to meet many of the formal and practical requirements for deep structure recovery, but at present not enough is known about its operating characteristics to assess its adequacy as a formalism for perceptual models. Augmented recursive transition network grammars, to which we now turn, can satisfy (a) - (c), have other desirable psychological and formal properties, and have the additional advantage of being practical and efficient.

III. The Augmented Recursive Transition Network

The idea of a transition network parsing procedure for natural language was originally suggested by Thome, Bratley, and Dewar (23) and was subsequently refined in an implementation by Bobrow and Fraser (3). Woods (26, 27) has also presented a transition network parsing system which is more general than either the Thorne _et al_. or Bobrow-Fraser systems. The discussion below is based on the Woods version. Since a detailed description is already available we present here only a brief outline of the grammatical formalism and then focus on the manner in which this formalism can be used to express perceptual and linguistic regularities.

At the heart of the augmented recursive transition network is a familiar finite-state grammar (5) consisting of a finite set of nodes (states) connected by labelled directed arcs. An arc represents an allowable transition from the state at its tail to the state at its heau, the label indicating the input symbol which must be found in order for the transition to occur. An input string is accepted by the grammar if there is a path of transitions which correspond to the sequence of symbols in the string and which lead from a specified initial state to one of a set of specified final states. Finite state grammars are attractive from the perceptual point of view because they process strings strictly from left to right, but they have well-known inadequacies as models for natural languages (4). For example, they have no machinery for expressing statements about hierarchical structure.

This particular weakness can be eliminated by adding a recursive control mechanism to the basic strategy, as follows: all states are given names which are then allowed as labels on arcs in addition to the normal input-symbol labels. When an arc with a state-name is encountered, the name of the state at the head of the arc is _pushed_ (saved on the top of a push-down store), and analysis of the remainder of the input string continues at the state named on the arc. When a final state is reached in this new part of the grammar, a _pop_ occurs (control is returned to the state removed from the top of the push-down store). A sentence is said to be accepted when a final state, the end of the string, and an empty push-down store are all reached at the same time. Note that with this elaboration of the basic finite-state mechanism, we have produced a formalism that can easily describe context-free languages as well as regular languages with unbounded coordinate structures. The structural description provided for a sentence by this procedure is simply the history of transitions, pushes, and pops required to get through the string.

However, the finite-state transition network with recursion cannot describe cross-serial dependencies, so it is still inadequate for natural languages (21). The necessary additional power is obtained by permitting a sequence of actions and a condition to be specified on each arc. The actions provide a facility for explicitly building and naming tree structures. The names, called registers, function much like symbolic variables in programming languages: they can be used in later actions, perhaps on subsequent arcs, to refer to their associated structures. A register is said to contain the structure it names, and the actions determine additions and changes to the contents of registers in terms of the current input symbol, the previous contents of registers, and the results of lower-level computations (pushes). This means that as constituents of a sentence are identified, they can be held in registers until they are combined into larger constituents in other registers. In this way deep structural descriptions can be fashioned in registers essentially independently of the analysis paths through the transition network.

Conditions furnish more sensitive controls on the admissibility of transitions. A condition is a Boolean combination of predicates Involving the current Input symbol and register contents. An arc cannot be taken if its condition evaluates to false (symbolized by NIL), even though the current input symbol satisfies the arc label. This means first, that more elaborate restrictions can be imposed on the current symbol than those conveyed by the arc label, and second, that information about previous states and structures can be passed along in the network to determine future transitions. This makes it possible for similar sections of separate analysis paths to be merged for awhile and then separated again — a powerful technique for eliminating redundancies and simplifying grammars. The condition predicates and the arc actions can be arbitrary functions in LISP notation, although we have developed a small set of primitive operations, described below and in (26, 27), which seem adequate for most situations. In these primitive actions and predicates, atomic arguments denote registers; parenthetic expressions are forms to be evaluated.

In order to be able to refer to the current input symbol in conditions or actions, a special register, named *, has been provided. More properly, this register always contains the constituent that enabled the transition; usually this is the input symbol, but for actions on a push arc (which are usually executed after the return from the lower level), * contains the structural description of the phrase identified In the lower computation. This phrase is determined when a special type of arc, a pop arc, is taken from a final state at the lower level (final states are distinguished by the existence of pop arcs).

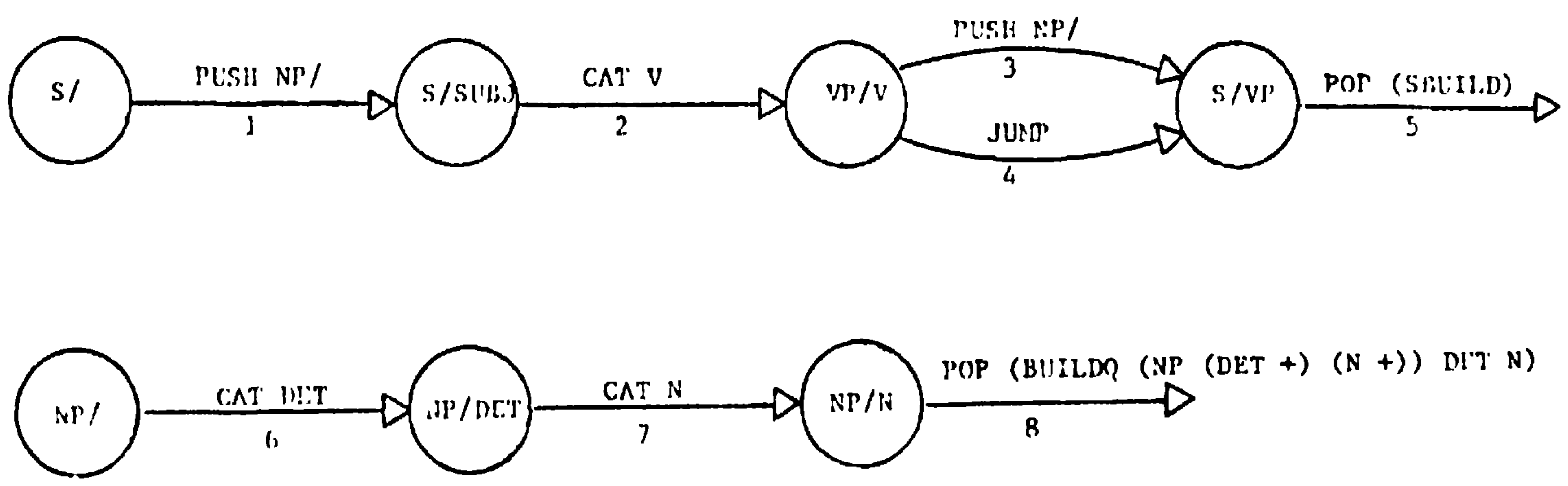
The recursive transition network, with all of these additions, Is called an augmented recursive transition network; it is easy to show that it has the generative power of a Turing machine. To demonstrate more concretely how the transition network works, we give a simple example. Figure 1 shows a transition network grammar that will recover deep structures for simple transitive and intransitive sentences, such as [6] and [7]:

[6] The man kicked the ball.

[7] The ball fell.

The top of the figure shows the organization of paths in the network. States are represented by circles with the state name inside. The state-names are purely mnemonic, serving to indicate the constituent being analyzed (to the left of the slash) and how much of that constituent has been identified so far. Each arc specifies what will allow the transition and has a number denoting the condition and actions in the table below. We mentioned above three kinds of arcs: ordinary input symbol arcs, push arcs, and pop arcs. To distinguish these arcs from each other and from other arc types, each arc has an explicit type-indicator. Thus, PUSH NP/ specifies that arc 1 is a push arc and that control is to pass to state NP/. POP (SBUILD) indicates that arc 5 Is a pop arc, and the structure to be popped (that is, placed in * at the next higher level) is the value of the function SBUILD. Figure 1 includes two new types: a CAT arc (arc 2) does not require a specific input symbol, but requires that the word be marked in the dictionary as belonging to the specified lexical category. A JUMP arc (arc 4) is a very special arc that allows a transition in the grammar,

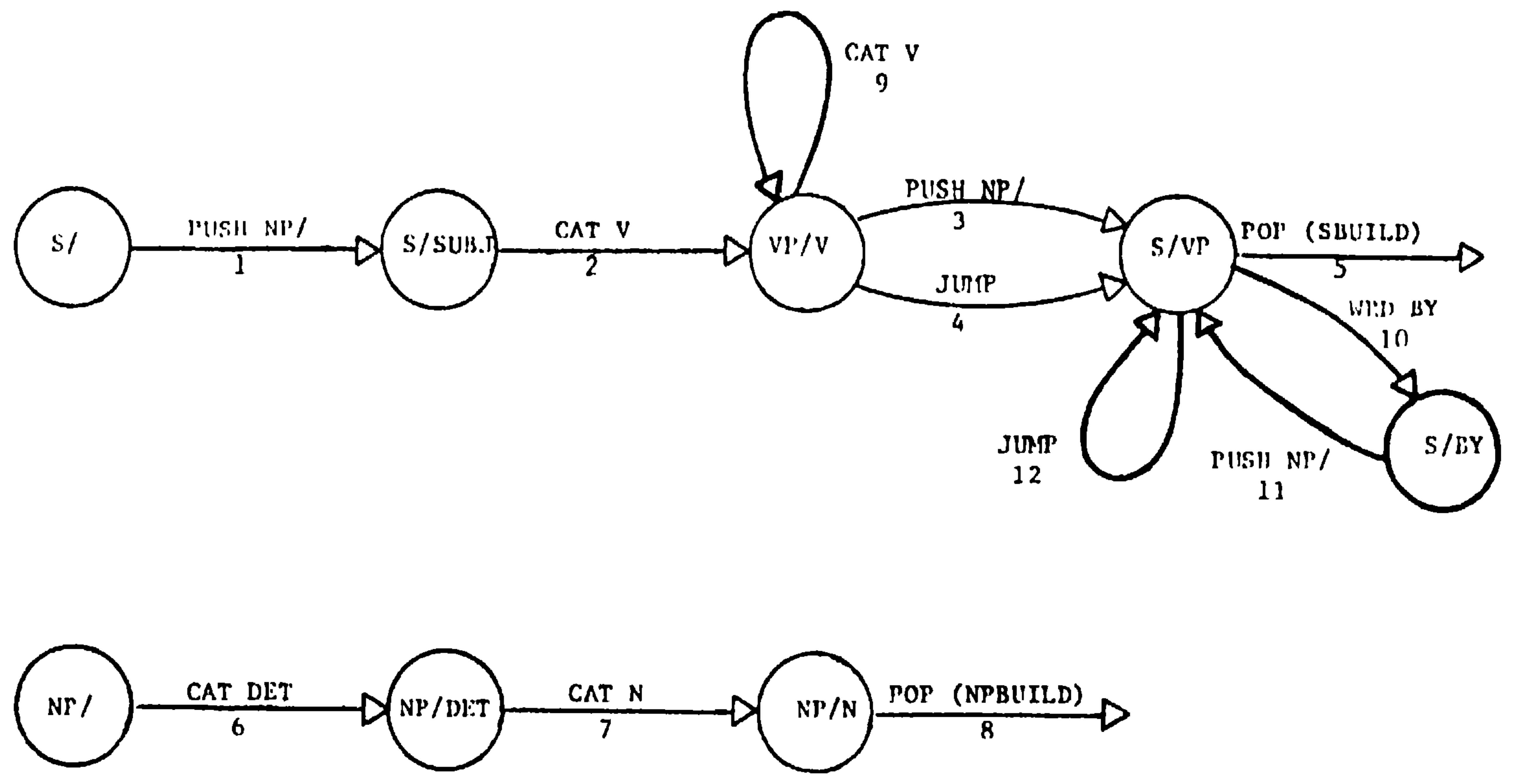
Figure 1: A Simple Transition Network Grammar



Arc	Condition	Actions
1	T	(SETR SUBJ *)
2	(AND (GETF TNS) (SVAGR SUBJ) (GETF PNCODE)))	(SETR TNS (GETF TNS))
3	(TRANS V)	(SETR OBJ *)

Arc	Condition	Actions
4	(INTRANS V)	
5	T	
6	T	(SETR DET *)
7	T	(SETR N *)
8	T	

Figure 2: Arcs Required for Passives



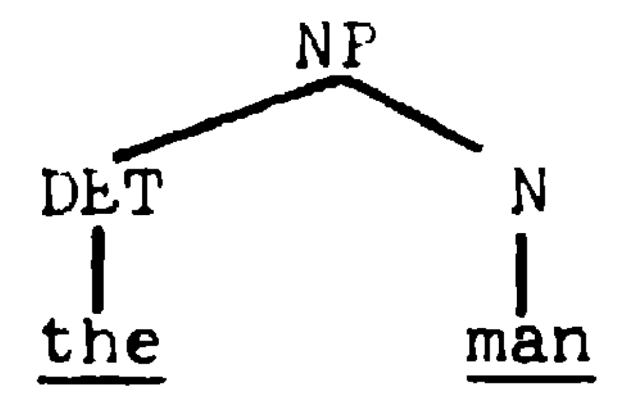
Arc	Condition	Actions
4	(OR (INTRANS V) (FULLR OBJ))	
5	(FULLR SUBJ)	
9	(AND (GETF PASTPART) (PASSIVE *) (WRD BE V))	(SETR OBJ SUBJ) (SETR SUBJ NIL) (SETR V *)

Arc	Condition	Actions
10	(NULLR SUBJ)	
11	T	
12	(NULLR SUBJ)	(SETR SUBJ (BUILDQ (NP (PRO SOMEONE))))

with possible actions, without advancing the input string — it is useful for by-passing optional grammar elements.

Let us trace the analysis of sentence [6] using this grammar (Figure 3 shows the trace as it is printed out by the program). The starting state is, by convention, the state labelled S/. The only arc leaving S/ is a push for a noun-phrase, so without advancing the input string, we switch to NP/. Since the, the current input symbol, is in the category DET and since the condition for arc 6 is trivially true, we can take arc 6, executing the action (SETR DET *). SETR is a primitive action that places the structure specified by its second argument (in this case, the current input word, denoted by *) in the register named by its first argument (DET). Thus after following arc 6, the register DET contains the, and we continue processing at state NP/DET, looking at the word man. We are permitted to take arc 7, saving man in the register N, and arrive at the final state NP/N. We take the POP arc, which defines the phrase to be returned. BUILDQ Is a primitive action that takes as its first argument a tree fragment with some nodes denoted by the symbol +. These nodes are replaced by the contents of the registers specified as the remaining arguments, in left-to-right order. Thus the value returned by arc 8 will be the structure (NP (DET the)(N man)), which is a labelled bracketing corresponding to the tree (8):

(8)



This structure is returned in the register * on arc 1, where the action (SETR SUBJ *) places it in the register SUBJ. We move on to state S/SUBJ, looking at the word kick.

Kick satisfies the label on arc 2, so the condition is evaluated, checking the inflectional features in the dictionary entry for kick. The predicate (GETF TNS) verifies that the verb is a tensed form (as opposed to a participle), and SVAGR ascertains that the person-number code of the verb agrees with the noun-phrase stored in the register SUBJ. Since the condition is true, the transition is permitted and the action is executed, setting the register TNS to the value of the feature TNS (in this case it would be PAST) and saving the verb in V. At state VP/V, we have a

Figure 3: Trace of an Analysis

Sentence: The man kicked the ball.

```
STRING = (THE MAN KICKED ThE BALL)
ENTERING STATE S/
ABOUT TO PUSH
 ENTERING STATE NP/
 TAKING CAT DET ARC

STRING = (MAN KICKED THE BALL)
 ENTERING STATE NP/DET
 TAKING CAT N ARC

STRING = (KICKED THE BALL)
 ENTERING STATE NP/N
 ABOUT TO POP
ENTERING STATE S/SUBJ
TAKING CAT V ARC

STRING - (THE BALL)
ENTERING STATE VP/V
STORING ALTARC ALTERNATIVE 76869[a]
ABOUT TO PUSH
 ENTERING STATE NP/
 TAKING CAT DET ARC

STRING = (BALL)
 ENTERING STATE NP/DET
 TAKING CAT N ARC

STRING - NIL
 ENTERING STATE NP/N
 ABOUT TO POP
ENTERING STATE S/VP
ABOUT TO POP
SUCCESS
10 ARCS ATTEMPTED
195 CONSES[b]
1.8869999 SECONDS[c]
PARSINGS:[d]
S NP DET THE
     N MAN
  AUX TNS PAST
  VP V KICK
     NP DET THE
        N BALL
```

a. The alternative analysis path starting with arc 4 is saved.

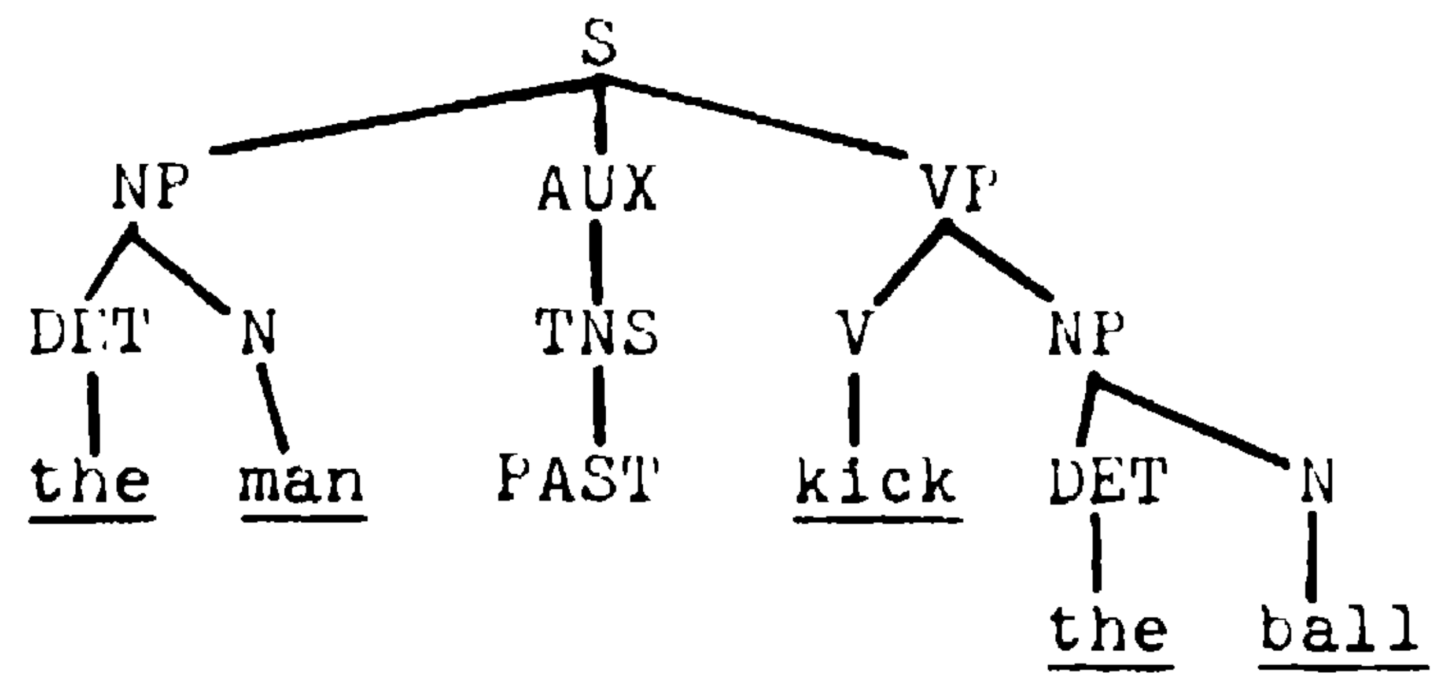
b. Number of memory words used.

c. Processing time required.

d. The recovered deep structure.

choice of two arcs. Are j is a push for an object noun-phrase, which we can take since (TRANS V) is true, that is, since the verb in V (kills) is marked transitive in the dictionary. We execute the push, identify the noun-phrase the ball, and save it in the register 0BJ. At S/VP we pop the value of SBU1LD, a function which applies a complicated BUILDQ to the registers SUBJ, TNS, V, OBJ, building the tree [9]. Notice that at this point we have exhausted the input string, achieved a final state, and emptied the push-down stack. Thus the sentence [6J is accepted by the grammar, and its deep structure is the structure returned by the final POP.

[9]



Sentence [7] is processed in the same way, except that arc 4 is taken instead of arc 3, since fell is marked intransitive. Hence, the resulting structure does not have the object NP node.

For these two examples and, indeed, for all sentences in the language of this grammar, the structure returned by the final POP directly reflects the history of the analysis -- the surface structure — but this need not be the case. As a second illustration, we extend the grammar to deal with passive sentences, such as [10J:

[10] The ball was kicked by the man.

We must add one new state, S/BY, a new arc to state VP/V and two new arcs to state S/VP. In addition, we must change the conditions on arcs 4 and 5. Figure 2 shows the new grammar, with new arcs in boldface and with only new and changed conditions and actions. For sentence [10] the new grammar works as follows: the ball Is recognized as a noun-phrase and placed in SUBJ. Was passes the condition on arc 2, so PAST Is stored in TNS and be is placed in V (as part of the category checking operation, the inflected form was is replaced in * by its root). At this point in the sentence, we do not know if be is a passive marker or a main verb as IE [11].

[11] The ball was a sphere.

We make the assumption that it is a main verb, with the understanding that later Information might cause us to change our minds and possibly rearrange the structure we have built. At state VP/V we find that we have indeed made a mistake. We first attempt the arc 9 transition. We are looking at kicked, the past participle of a passivizable verb, and be is in V, so we can make the transition: the contents of SUBJ (the ball) are moved to OBJ and SUBJ is emptied (a register containing NIL is considered void). Then kick replaces be in V, and we re-enter state VP/V, looking at the word by.

By is not a verb, so arc 9 is disallowed. Kick is transitive, so we try pushing for a noun-phrase, but since by is not a determiner, the push is unsuccessful. Arc 4 has been modified so that It can be taken if the verb Is transitive but the object register has already been filled (the predicate FULLR is true Just in case the indicated register is non-empty), and we can therefore JUMP to S/VP.

At S/VP we cannot take arc 5 because we have no subject, so we try arc 10, a WRD arc. This arc type corresponds to the original finite-state grammar arc-label, a symbol which must literally match an input word. Arc 10 specifies WRD BY and matches the current word, so the transition is allowed (NULLR is true when FULLR is NIL). At this point in the sentence, the only way we could not have a subject is if we had followed the passive loop. We therefore look for the deep subject of the sentence in a by-phrase: we take arc 11, put the man in SUBJ, and return to S/VP, from which we pop. The resulting structure is Identical to [9] -- we have undone the passive transformation. If the agent phrase had been omitted in L10], we would have taken arc 12 instead of the path through S/BY. Arc 12 is a JUMP that inserts the pronoun someone in SUBJ just in case there is no other way to get a subject.

These simple examples have illustrated the notation and underlying organization of the augmented recursive transition network. They have also demonstrated that transition network grammars can perform such transformational operations as movement, deletion, and insertion in a straightforward manner. We are now ready to examine the way in which transition network grammars can model performance data.

IV. The Formalization of Perceptual Strategies

Bever (1) has surveyed the results of many psycholinguistic experiments and has inferred from the data that human beings use a small number of perceptual strategies in processing sentences. Some of these are corollaries of more general cognitive strategies and have observable reflexes in other areas of perception, while others are peculiar to language performance. As a set, these strategies account in part for the relative perceptual complexity of sentences and for some of the patterns of observed perceptual errors. In this section, we show how these strategies can be naturally represented in transition network grammars.

The dependent variable in a majority of psycholinguistic studies has been the difficulty subjects experience in processing sentences, as indicated for example by response latencies, recall errors, and the impact of various disturbances on comprehensibility. Thus the ultimate validation of transition network models will depend to a large extent on the correlation between experimentally observed complexity and complexity as measured in the transition network. There are several ways of defining a complexity metric on the network. We could count the total number of transitions taken in analyzing a sentence, the total number of structure building actions executed or even the total number of tree-nodes built by these actions. We could also use the amount of memory space or computing time required for a sentence in a particular implementation of the transition network parser (e.g., the number of conses (memory cells) or seconds indicated in Figure 3). Of coarse, most intuitive measures of complexity are highly intercorrelated and lead to the same predictions, so our choice can be somewhat arbitrary. We will say that the complexity of a sentence is directly proportional to the number of transitions made or attempted during the course of its analysis.

With this definition the complexity of a sentence depends crucially on the order in which the network is searched for a successful path, although its acceptability by the grammar is independent of the search-order. Unless special mechanisms are invoked, the arcs leaving a state-circle are tried in clockwise order, starting from the top. Thus in Figure 2, arc 5 is attempted before arcs 10 and 12. If an attempted arc turns out to be permitted, then the remaining, untried arcs leaving the state are held in abeyance on a list of alternatives, and the legal transition is made. If the path taken is subsequently blocked, alternatives are removed from the front of the list and tried until another legal path is found. As a result of this depth-first search, an ambiguous sentence will initially provide only one analysis; the other analyses are obtained by simulating blocked paths after successes.

A. The Relations Between Clauses

Since sentences are frequently composed of more than one clause, the native speaker must have a strategy for deciding how the component clauses of a sentence are related to each other (e.g., which is the main clause, which are relative clauses, and which are subordinate). Bever propounds that "the first N..V..(N) clause... is the main clause, unless the verb is marked as subordinate" (1, Strategy B, p. 294), and points out that a sentence is perceptually more complicated whenever the first verb is not the main verb, even if it is marked as subordinate.3 Thus, sentences with preposed subordinate clauses [12b] are, according to this hypothesis, relatively more difficult than their normally ordered counterparts [12a]:

[12] a. The dog bit the cat because the food was gone.

b. Because the food was gone, the dog bit the cat. (=Dever's [2^a-b]).

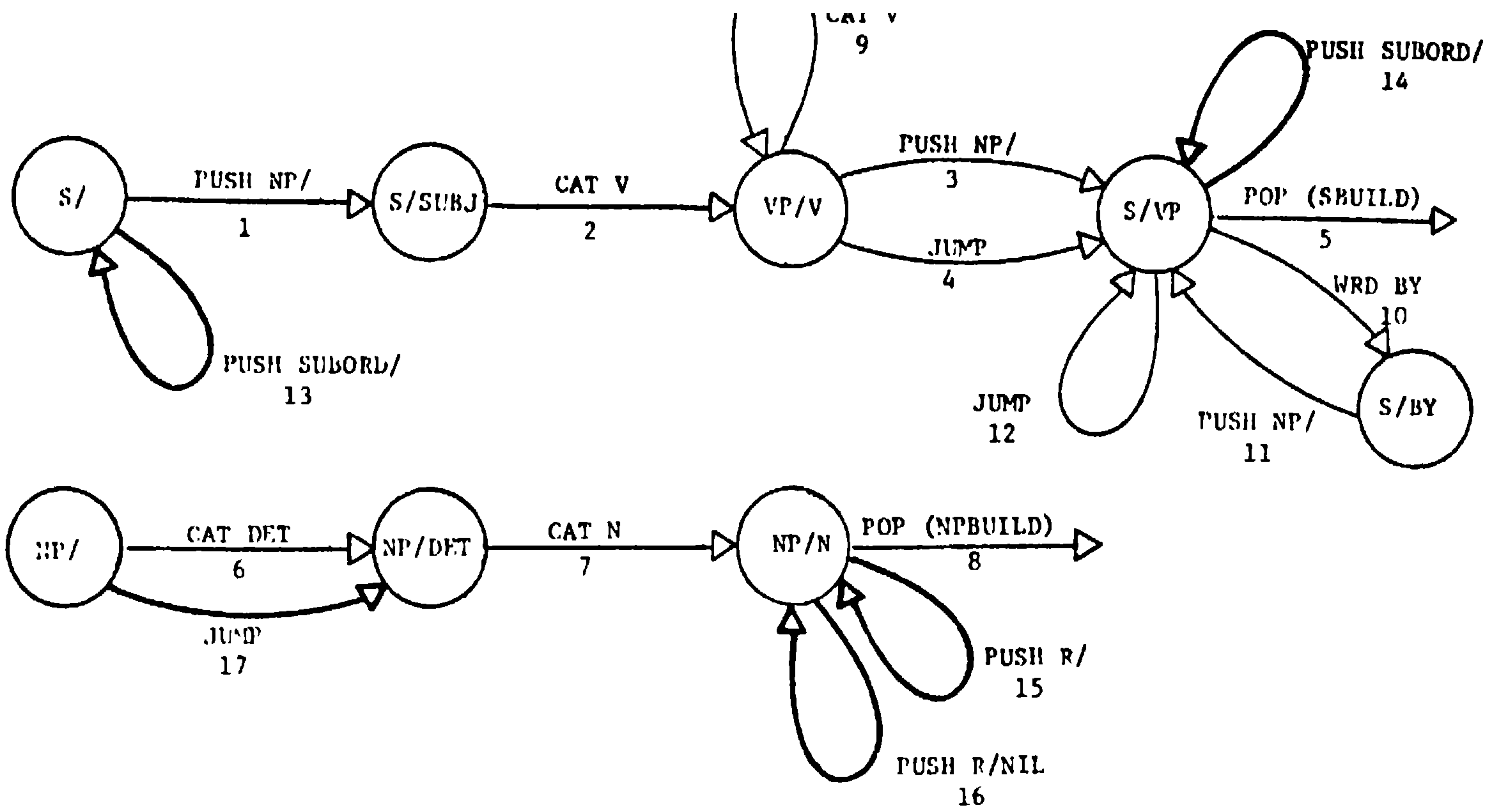
And in cases where the verb is subordinate but not marked as such, this strategy can lead to serious perceptual errors. Bever reports that subjects had much more difficulty understanding sentences like [13a], where there is an illusory main verb and sentence (underlined), than [13b], even though both sentences, being center-embedded, are exceedingly difficult:

[13] a. The editor authors the newspaper hired liked laughed.

b. The editor the authors the newspaper hired liked laughed. (=Bever's [27a-b])

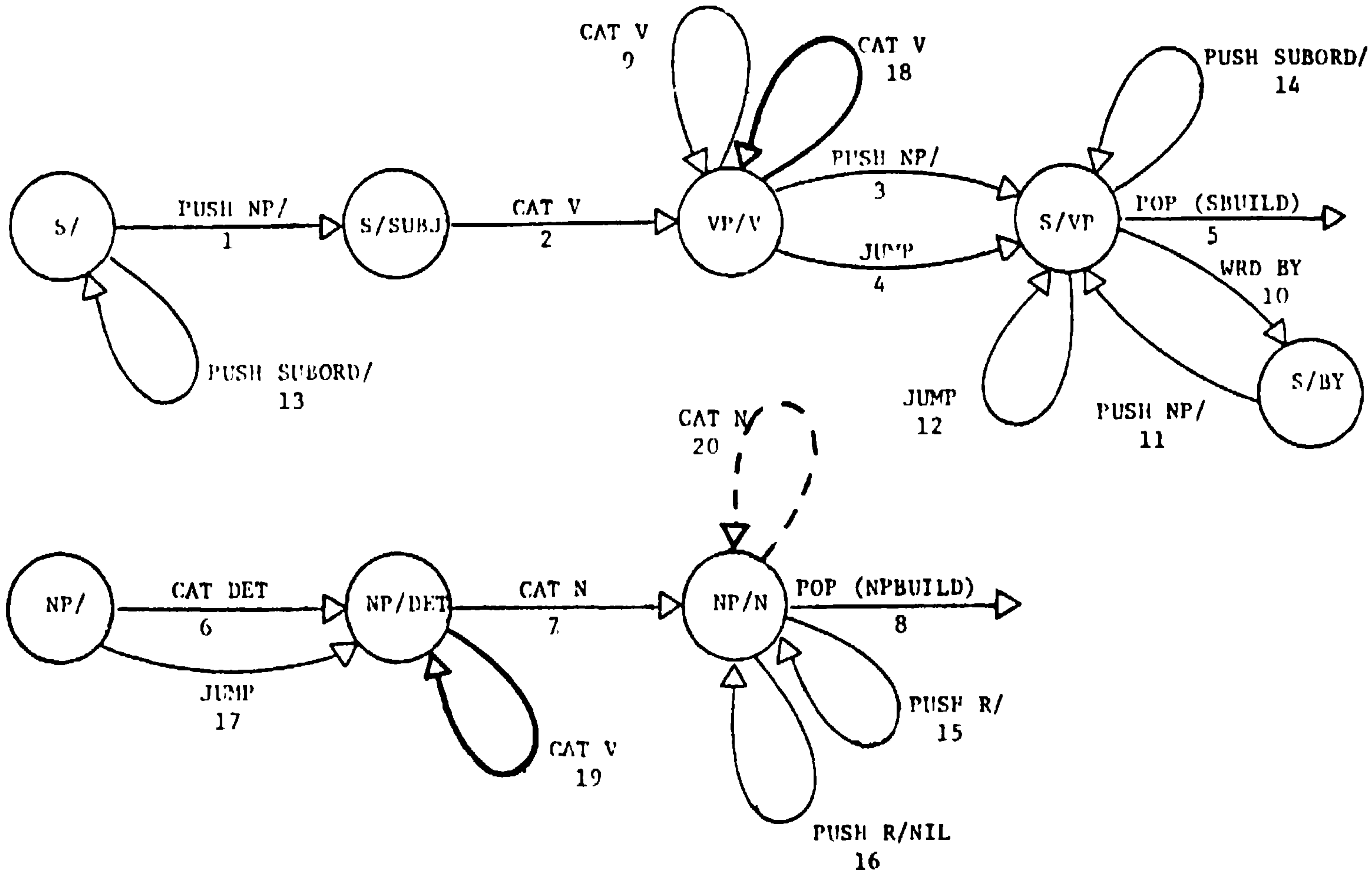
The modifications to our transition network shown in Figure 4 can account for these facts. We have added two arcs

Figure A: Clausal Relationships



Arc	Condition	Actions	Arc	Condition	Actions
13	(NULLR SUBORD)	(SETR SUBORD *)	16	T	(SENDR WH (NPBUILD)) (ADDR REL *)
14	(NULLR SUBORD)	(SETR SUBORD *)			
15	(CAT RELPRO)	(SENDR WH (NPBUILD)) (ADDR REL *)	17	T	

Figure 5: Progressives and Prenominal Modifiers



Arc	Condition	Actions	Arc	Condition	Actions
18	(AND (GETF PRESPART) (WRD BE V))	(SETR V *) (ADDR TNS (QUOTE PROG))	19	(GETF PRESPART)	(ADDR NMODS *)
			20	(GE (NLIKE *) (NLIKE N))	(ADDR NMODS N) (SETR N *)

at the S level to look for subordinate clauses, a simple transition sequence (not shown) analyzes and builds the appropriate structure for them. Also, we have expanded states NP/ to allow null determiners, and NP/N to look for relative clauses. With this grammar, four more arcs, 1, 6, 17, and 7, must be attempted for [12b] than for [12a]. For [12b], first arc 1 is tried, causing a push to NP/ where arcs 6, 17, and 7 are tried and fail. We back up to state S/ and take arc 13, eventually ending up with the appropriate structure (the complete sequence of attempted arcs is 1, 6, 17, 7, 13, SUBORD/ arcs (not shown), 1, 6, 7, 8, 2, 9, 3, 6, 7, 8, 14, 5). Note that we must still attempt arc 14, even though we know the condition will fail, because it is ordered before the pop arc, arc 5. For [12a], our first try at arc 1 takes us straight through to arc 14, where we pick up the subordinate clause, consider arc 14 again, and then pop at arc 5 (sequence = 1, 6, 7, 8, 2, 9, 3, 6, 7, 8, 14, SUBORD/arcs, 14, 5)

The difference between [13a] and [13b] is equally well accounted for. Arc 15 looks for a relative clause on the noun-phrase, given that there is a relative pronoun following the noun. The arc has two new actions, SENDR and ADDR. Registers are subject to the control of the push-down recursion mechanism, so that when a push is executed, the registers' contents at the upper level are saved on the stack along with the actions to be executed upon return, and at entry to the lower-level, the registers are all empty. Upon popping, the upper level registers are restored. SENDR is a very special action: it can only appear on a PUSH arc, and it is the only action executed <u>before</u> pushing. It causes structures computed at the upper level to be placed in registers at the lower level. Thus the action (SENDR WH (NPBUILD)) causes the noun-phrase so far identified to be placed in the WH register at state R/, the beginning of the relative clause network (not shown). Based on the internal structure of the relative clause, the R/ network decides whether the relativized noun-phrase in WH is to be interpreted as the subject or object, analyzes the clause using parts of the S/ and NP/ networks, and returns the appropriate structure. (ADDR REL *) causes this structure to be <u>add</u>ed on the right of the previous contents of REL, so that a sequence of relative clauses can be processed by looping through arc 15.

In [13a-b], however, there is no relative pronoun, so we cannot take arc 15. For both sentences, a successful analysis requires that we push to state R/NIL (arc 16), the section of the relative clause grammar designed to analyze relatives with missing relative pronouns. But before we get to arc 16, we pop via arc 8 to state S/SUBJ. In [13a], the Input word at this point is <u>authors</u>, a possible verb, so we can take arc 2 to state VP/V. We continue on until we try to pop at arc 5 without having consumed the input string (the current word is <u>hired</u>), and by the time we have backed up all the way to the appropriate arc 16, we have attempted seventeen arcs erroneously (sequence=1, 6, 7, 8, 2, 9, 3, 6, 7, 8, 14, 5, 10, 12, 15, 16, blocked R/NIL arcs, 17, 7, 4, 15, 16, R/NIL arcs, 8, 2, 9, 3, 6, 17, 7, 4, 14, 5). For [13b], since the is not a verb, we are blocked at state S/SUBJ, and we arrive at arc 16 having only attempted three wrong arcs (sequence -1, 6, 7, 8, 2, 15, 16, R/NIL arcs, 8, 2, 9, 3, 6, 17, 7, 4, 14, 5). Inside the relative clause grammar, the noun phrase <u>authors</u> in [13a] requires an extra transition at arc 17, so the net difference between the two sentences is fifteen arcs, not counting the blocked R/NIL arcs in [13a], a difference clearly in line with empirical perceptual complexity.

We have thus expanded our simple grammar to accept and provide deep structures for a variety of constructions. Our grammar has the same formal power to describe these structures as a transformational grammar, but we have been able to arrange the analysis path so that complexity in our model corresponds to perceptual complexity, as stated by Bever's Strategy B. We have taken advantage of the fact that, unlike the ordering of transformations, the order of arcs can be freely changed, radically altering the amount of computation required for particular sentences, without affecting the class of acceptable sentences.

B. Functional Labels

A major task in sentence comprehension is the determination of the functional relationships of constituents within a single clause, of deciding who the subject is, what the action Is, etc. Bever suggests a simple strategy for assigning functional labels based on the left-to-right surface order of constituents: "Any Noun-Verb-Noun (NVN) sequence within a potential internal [deep structure] unit in the surface structure corresponds to 'actor-action-object'" (1, Strategy D, p.298). Bever cites several perceptual studies involving sentences for which this strategy Is misleading, and in all cases, these sentences were more difficult to respond to than control sentences for which strategy D was appropriate.

There is very good evidence that passive sentences are more difficult to process than corresponding actives, in the absence of strong semantic constraints. Given strategy D, this follows from the fact that the surface order of passives is object-action-actor. Similarly, progressives [14a] have been found to be significantly easier to comprehend than superficially identical participial constructions [14b] (18).

[14] a. They are fixing benches.

b. They are performing monkeys. (=Bever's [31 a-b])

According to strategy D, <u>performing</u> is initially accepted as the main verb, until the spurious direct object <u>monkeys</u> is encountered,[4] at this point the labels must be switched around.

Bever explaine these processing difficulties in terms of the amount of relabeling that is required, given that strategy D can lead to errors. This translates into the proposition that relative complexity is measured by the degree to which constituents are shifted in registers, since assigning a constituent to a register is the transition network analog of functional labeling. Indeed, Figure 2 shows that SUBJ is reset twice more for passives than for actives, while in Figure 5 participial sentences require one extra register assignment (NMODS). However, we have defined complexity in terms of the number of arcs attempted, and we now show that this measure can also account for the experimental results.

Figure 2 contains the arcs necessary for passive sentences. Simple active [6] and passive [10] sentences are treated identically until state VP/V is reached. Arc 9 is attempted for both of them and is taken for the passive, returning to VP/V. 9 is attempted again but fails, and then twelve additional arcs are tried before the successful final pop is executed. Since only six additional arcs are attempted for the active, the difference in favor of the relative complexity of the passive is six. (The difference is seven for the more complicated grammar in Figure 5.)

Figure 5 gives the necessary modifications for the progressive and participial constructions (in bold face). Arc 18 can be taken only if the current word is a present participle and the previously identified main verb is <u>be</u>. The actions put the new verb in V and mark TNS as progressive. Arc 19 simply adds an identified participle to NMODS, where the function NPBUILD will find it. The analysis of [14a] is simple: at state VP/V, the current word will be <u>fixing</u> and be will be in V, so that arc *lb* can be taken. Since <u>fix</u> is transitive, <u>benches</u> will be identified as the direct object, and the pop at arc 5 will be successful. [14b] involves considerably more effort. At VP/V, arc 18 will also be taken but arc 3 is ruled out with <u>perform</u> in V (see foot note 1). Before returning to arc 3 with be in V, arcs 1, 11, 5> 10, and 12 will be tried, and additional arcs will be attempted in deriving the correct participial analysis (we assume that <u>be</u> is marked transitive).

Thus the functional-relabelling and the attempted-transitions explanations account equally well for the experimental observations. At present we have no firm empirical basis for choosing one complexity measure over the other; we must find crucial sentences where the measures make opposing predictions and let the data decide for us. So far, we have been unable to discover such sentences.

C. Prenominal Adjective Ordering

Another problem concerns the segmentation of superficial sequences of words into structural units. Where does a noun-phrase begin, for example, and where does it end? That these are not trivial questions is illustrated by [15a-b], where the role of <u>marks</u> is unclear until the whole sentence has been processed.

[15] a. The plastic pencil marks easily.

b. The plastic pencil marks were ugly. (=Bever's [66a-b])

Of course, no matter what perceptual strategy is involved in making these decisions, the transition network will continue trying alternative paths until it arrives at the correct segmentation, but an appropriate strategy would make the analysis more efficient. Bever suggests that in recognizing the end of a noun-phrase, native speakers use a strategy which also accounts for the anomalies in such pairs as (without contrastive stress):

[16] a. The red plastic box...

b. *The plastic red box...

c. The large red box...

d. *The red large box...
(=Bever's [67a-d])

He cites the theories of Martin (17) and Vendler (25) which essentially claim that the more "nounlike" an adjective is[5], the closer to the noun it must be placed.Thus the anomalies in [16] are accounted for if we assume that plastic is more nounlike than red and red is more nounlike than large. Although the notion nounlike is not made very precise, Bever gives heuristic arguments that these assumptions are correct. He then postulates that the end of a noun phrase is signalled by a word which is less nounlike than preceding words (1, Strategy E, p. 323). Since large is less nounlike than red, the initial noun phrase in [l6d] must be the red.

This constraint is difficult to express in traditional transformational formalisms but is quite directly representable in the transition network. It not only makes the transition network more congruent with performance data but also helps to rule out the anomalies in [16]. Assuming that nounlike is well-defined and that all potential nouns (including adjectives) are in category N and have their nounlike-ness marked in the lexicon, the dashed arc in Figure 5 is the necessary addition to the network. Arc 20 is attempted before the pop from NP/N. If the nounlike-ness of the current word is greater than or equal to that of the word in N, then the word in N is not the head of the noun-phrase. We add this word to the list of modifiers in NMODS, and place the current word in N, as a new candidate for head noun. We continue looping until we find a word that is less nounlike than the head, marking the end of the noun-phrase. This procedure will accept [16a,c] but reject [16b,d] except in constructions along the lines of [17]. In [17] the adjectives are accepted only because they can be analyzed in separate noun-phrases:

[17] I like the plastic red boxes are made of.

V. The Justification of Transition Network Models

In the preceding sections we illustrated the simple way in which transition network grammars can express some of Bever's perceptual strategies. The transition network analyzes strings in essentially linear order, and the grammatical notation is flexible enough so that grammars can be devised to fit wide ranges of performance facts. However, to Justify the effort needed to simulate experimental data with network models, we must show that the resulting grammars offer substantial advantages compared to informal verbal interpretations, such as Bever's. In this section we argue that these grammars are both conceptually and empirically productive: they lead to new theoretical questions, and they suggest new lines of experimentation, predicting specific outcomes. To the extent that the predictions of a particular grammar are confirmed, that grammar is validated as a model of the psychological processes involved in sentence comprehension.

The grammar shown in Figure 5, while only a small fragment of a complete English grammar, will suffice to exemplify the empirical implications of transition network models. It has been designed to account for the data underlying the perceptual strategies discussed above, but it also encompasses independent findings. The grammar mirrors the perceptual strategies Just so long as a depth-first search procedure is used to discover successful analysis paths. This search order implies that for truly ambiguous sentences, one interpretation will be recovered before the other; if required, the second interpretation can be recovered by simulating a failure and continuing the analysis. This is in line with the results of MacKay and Bever (15).and Foss et al. (9): MacKay and Bever found subjects to be aware that they arrived at one interpretation of an ambiguous sentence first and could even report what the first interpretation was. Foss et al. discovered that subjects tend to interpret ambiguous sentences in only one way; if the first interpretation is incompatible with the experimental context, they can usually go on to find another interpretation, although additional time is required. The search strategy underlying the Figure 5 grammar accounts for these results even though the experiments are not implicated in the perceptual strategies the grammar was designed to represent .

For ambiguous sentences within its scope, the grammar clearly predicts which interpretation should predominate. Other things being equal, the first interpretation will have essentially the same analysis as the less complex of two unambiguous sentences with the same surface structure. Thus in a replication of the Foss et al. experiment, the first analysis of [18a] should be the progressive, resembling [14a], while the participial deep structure [14b] should come out second. Subjects should first arrive at the interpretation paraphrased in [18b], rather than [18c]: