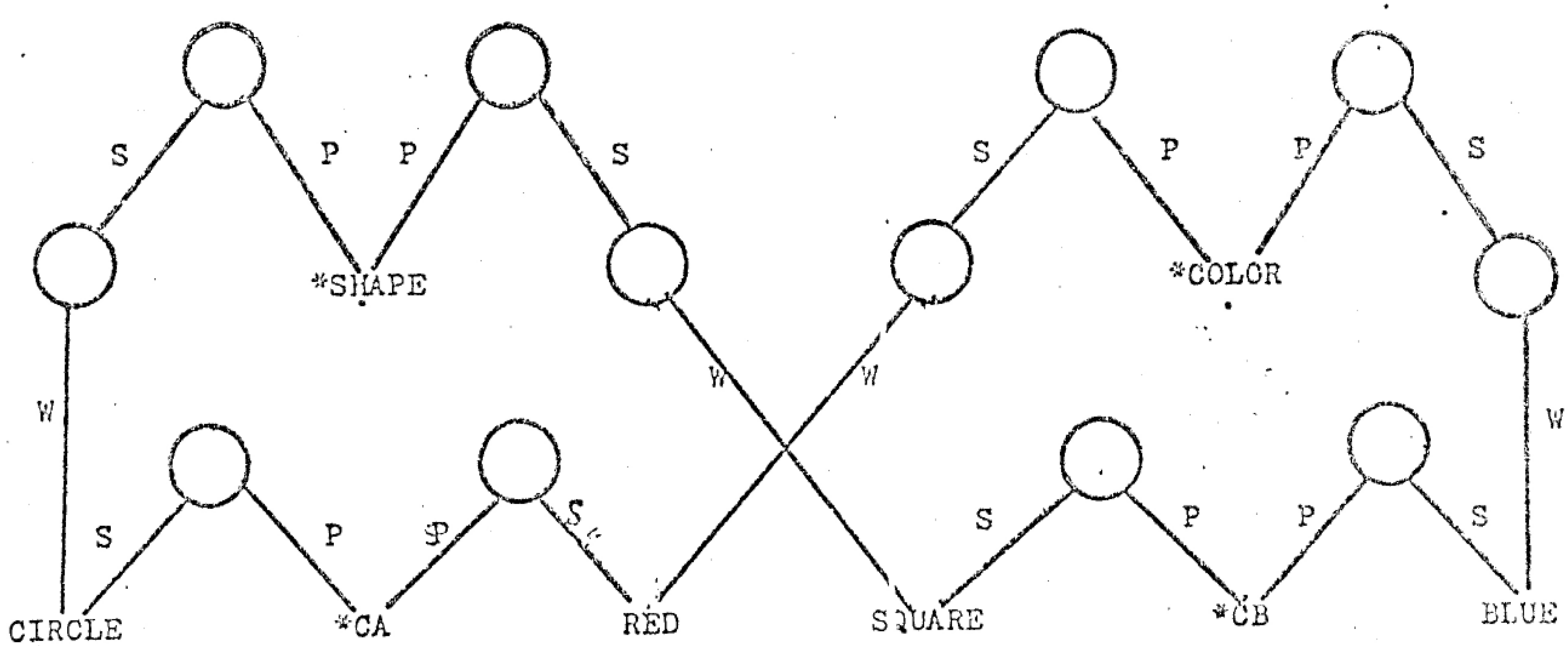


Figure 4. An example of a HAM structure encoding both categorical information and word class information

(Ideate red 1)
(Ideate square 2)
(Ideate above 3)
(Ideate circle 4)
(Out-of X 1)
(Out-of X 2)
(Out-of X 8)
(Objectify 8 Y)
(Relatify 8 3)
(Out-of Y 4)

The Network Grammars

Here the formalisms of LAS's network grammar will be described. These formalisms are intended to apply to any natural language. In illustration, the grammars for two test languages will be presented. These test languages will also be used to illustrate the SPEAK and UNDERSTAND programs to be described shortly. The first, GRAMMAR1, is a simple artificial grammar. The second, GRAMMAR2, is a more complex grammar for a subset of English. They are defined by the rewrite rules in Table 1. GRAMMAR1 was designed to be maximally different from English word order. The sentences of GRAMMAR1 are to be read as asserting the first noun-phrase has the relation specified by the last word to the second noun phrase. For purposes of readability, the words of these languages are English but they need not be. GRAMMAR1 is a finite language without recursion. In contrast, in GRAMMAR2 the NP element has an optional CLAUSE which can recursively call NP, generating a potential infinite embedding of constructions.

In both grammars, it is assumed that *above* and *below* are connected to the same idea as are *right-of* and *left-of*. The words differ in the assigment of their NP arguments to subject and object roles. Thus the difference between the word pairs is syntactic. This is indicated by having the words belong to two word classes RA and RB. Thus, UNDERSTAND with GRAMMAR2 would derive the same HAM representation in Figure 3 for the sentences *The red square is above the circle* and *The circle is below the red square*. It would have been possible to generate distinct representations for these two sentences. I think this would have been less psychologically interesting. Basically, the network grammar makes the inferences that *A below B* is equivalent to *B above A* and encodes the latter.

TABLE 1

The Two Test Grammars

GRAMMAR1

S	→	NP NP RA
		NP NP RB
NP	→	SHAPE (COLOR) (SIZE)
SHAPE	→	square, circle, et.
COLOR	→	red, blue, etc.
SIZE	→	large, small, etc.
RA	→	above, right-of

GRAMMAR2

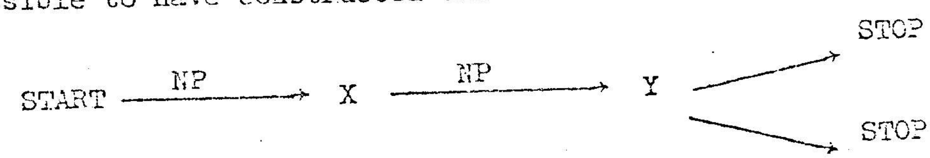
S	→	NP is ADJ
		NP is RA NP
		NP is RB NP
NP	→	(the,a) NP* CLAUSE
NP*	→	SHAPE
	→	ADJ SHAPE
CLAUSE	→	that is ADJ
		that is RA NP

TABLE 1 continued

R → below, left-of

CLAUSE → that is RB NP
SHAPE → square, circle, etc.
ADJ → red, big, blue, etc.
RA → above, right-of
RB → below, left-of

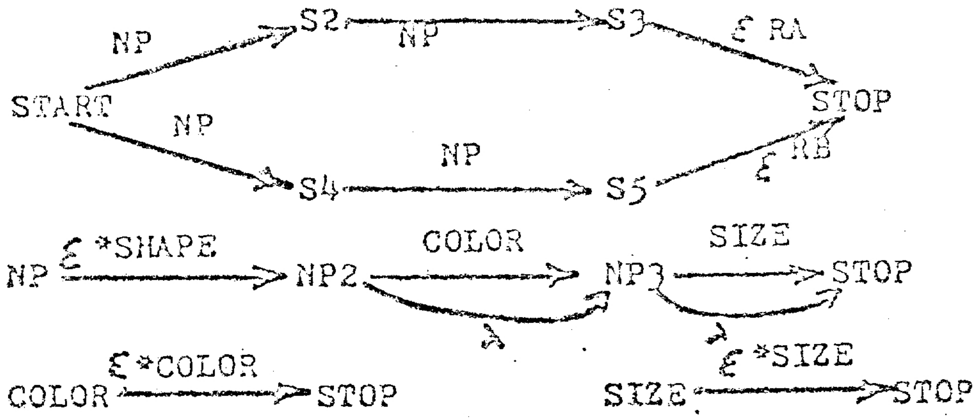
Figure 5 illustrates the parsing networks for the grammars. It should be understood that these networks have been deliberately written in an inefficient manner. For instance, note in GRAMMAR1 that there are two distinct paths in the main START network. The first is for those sentences with RA relations and the second for those sentences with RB relations. If a sentence input to UNDERSTAND has a RB relation, UNDERSTAND will first attempt to parse it by the first branch. The two noun phrase branches will succeed but the relation branch will fail. UNDERSTAND will have to back-up and try the second branch that leads to RB. This costly back-up is not really necessary. It would have been possible to have constructed the START network in the following form:



In this form the network does not branch until the critical relation word is reached. This means postponing until the end the assignment of noun phrases to subject and object roles in the representations of the sentence's meaning. The above network was not chosen because we wanted a more demanding test of the backup facilities of SPEAK and UNDERSTAND.

Table 2 provides a formal specification of the information stored in LAS's network grammars. A node either has a number of arcs proceeding out of it (1a) or it is a stop node (1b). In speaking and understanding LAS will try to find some path through the network ending with a stop node. Each arc consists of some condition that must be true of the sentence for that arc to be used in parsing (understanding) the sentence. The second element is an action to be taken if the condition is met. This action will create a piece of HAM conceptual structure to correspond to the meaning conveyed by the sentence at that point. Finally, an arc includes specification of the next node to which control should transfer after performing the action. An action consists of zero or more HAM memory commands (rule 3). A condition can consist of zero or more memory commands also (rule 4a). These specify properties that must be true of the incoming word. Alternatively, a condition may involve a push to an embedded network (rule 4b). For instance, suppose the structure in Figure 3 were to be spoken using GRAMMAR1. The START network would be called to realize the X is above Y proposition. The embedded NP network would be called to realize the X is red and X is square propositions. In pushing to a network two things must be specified--NODE, which is the embedded network and VAR, which is the memory node at which the main and embedded propositions intersect. The element t is rule 4b is a place-holder for information that is needed by the control mechanisms of the UNDERSTAND program. The three rules 6a, 6b, and 6c specify three types of arguments that memory commands can have. They can either directly refer to memory nodes, or refer to the current word in the sentence, or refer to variables which are bound to

Networks for Grammar1:



Networks for GRAMMAR2:

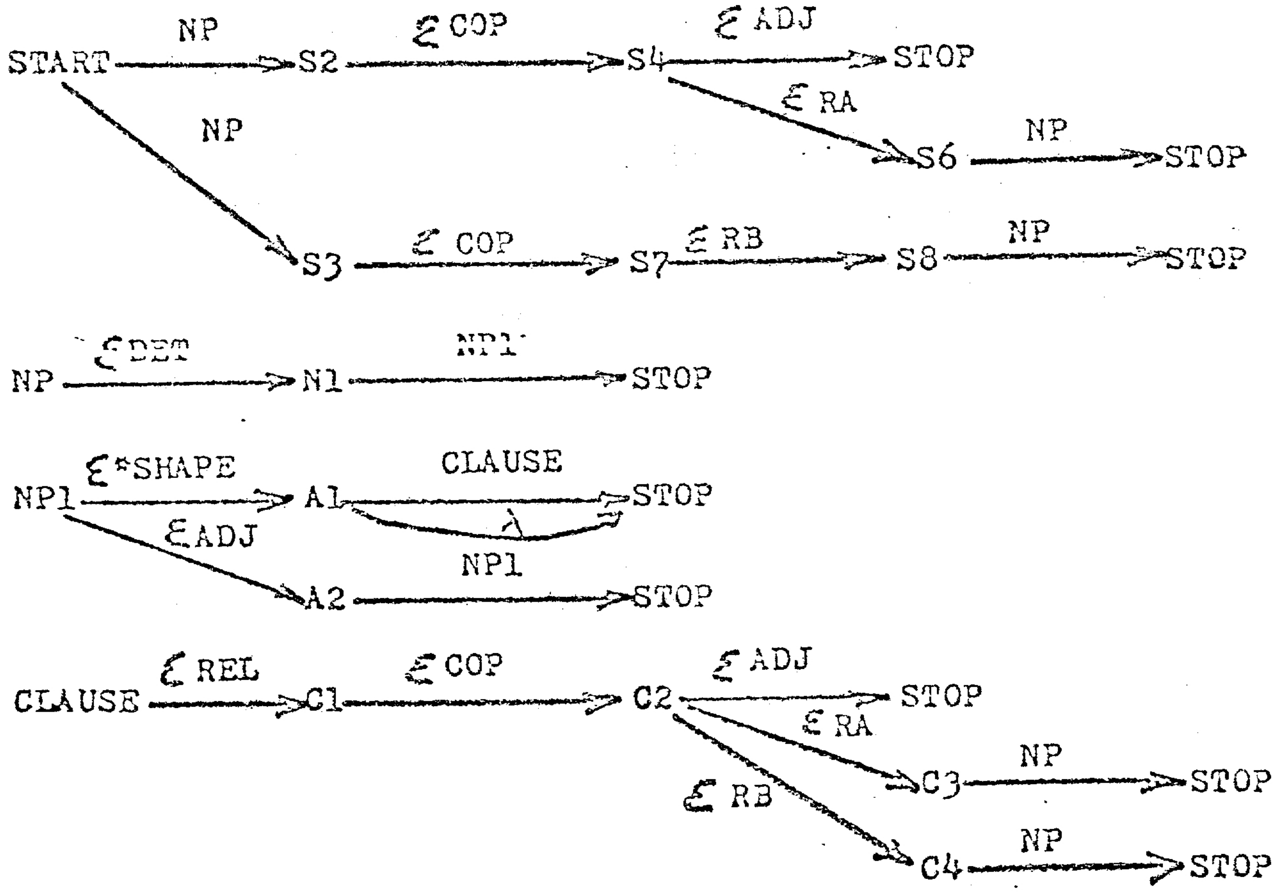


Figure 5. The network grammars used by LAS

memory nodes in the course of parsing.

TABLE 2

Formal Specification of the Network Grammar

NODE	→	ARC*	(1a)
	→	stop	(1b)
ARC	→	CONDITION ACTION NODE	(2)
ACTION	→	COMMAND*	(3)
CONDITION	→	(COMMAND*)	(4a)
	→	push VAR t NODE	(4b)
COMMAND	→	FUNCTION ARG ARG	(5)
ARG	→	memory node	(6a)
	→	word	(6b)
	→	X1, X2, X3, X4, X5	(6c)
FUNCTION	→	out-of, objectify, relatify, ideate	(7)

Table 3 provides the encoding of the network for GRAMMAR1.

Note that there tends to be a 1-1 correspondence between HAM propositions and LAS networks. That is, each network expresses just one proposition and calls one embedded network to express any other propositions. This correspondence is not quite perfect in GRAMMAR1 or GRAMMAR2, but as we will see, the grammars induced by LEARNMORE have necessarily a perfect correspondence.

These grammar networks have a number of features to commend them. SPEAK and UNDERSTAND use the same network for sentence comprehension and generation. Thus, LAS is the first extant system to have a uniform grammatical notation for its parsing and generation systems. In this way, LAS has only to induce one set of grammatical rules to do both tasks. Such networks are modular in two senses. First, they are relatively independent of each other. Second, they are independent of the SPEAK and UNDERSTAND programs that use them. This modularity greatly simplifies LAS's task of induction. LAS only induces the network grammars; the interpretative SPEAK and UNDERSTAND programs represent innate linguistic competences. Finally, the networks themselves are very simple with limited conditions and actions. Thus, LAS need consider only a small range of possibilities in inducing a network. The network formalism gains its expressive power by the embedding of networks. Because of network modularity, the induction task does not increase with the complexity of embedding.

It might be questioned whether it is really a virtue to have the same representation for the grammatical knowledge both for understanding and production. It is a common observation that children's ability to understand sentences precedes their ability to generate sentences. LAS would not seem to be able to simulate this basic fact of language learning. However, there may be reasons why child production does not mirror comprehension other than that different grammatical competences underlie the two. The child may not yet have acquired the physical mastery to produce certain words. This clearly is the case, for instance, with Lenneberg's (1962) anarthric child who under-

Table 3

The contruction of GRAMMAR1

```
(PUT 'DEFPROP 'NSUBR (GET 'PUT 'SUBR))
(PROGN
 (DEFPROP START PATH
  (((PUSH X1 T NP) ((OUT-OF X1 X5)) S2 )
   ((PUSH X1 T NP) ((OBJECTIFY X5 X1)) S4 )))
 (DEFPROP S2 PATH
  (((PUSH X2 T NP) ((OBJECTIFY X5 X2)) S3 )))
 (DEFPROP S3 PATH
 ((((IDEATE WORD X4) (OUT-OF WORD *RA)) ((RELATIFY X5 X4)) STOP
 (DEFPROP S4 PATH
  (((PUSH X2 T NP) ((OUT-OF X2 X5)) S5 )))
 (DEFPROP S5 PATH
((((IDEATE WORD X4) (OUT-OF WORD *RB)) ((RELATIFY X5 X4)) STOP )
 (DEFPROP NP PATH
 ((((IDEATE WORD X4) (OUT-OF X4 *SHAPE)) ((OUT-OF X1 X4)) NP2 ))
 (DEFPROP NP2 PATH
  (((PUSH X1 T COLOR) NIL NP3 )
   ( NIL NIL NP3)))
 (DEFPROP NP3 PATH
  (((PUSH X1 T SIZE) NIL STOP )
   (NIL NIL STOP)))
 (DEFPROP COLOR PATH
 ((((IDEATE WORD X4) (OUT-OF X4 *COLOR)) ((OUT-OF X1 X4)) STOP
 (DEFPROP SIZE PATH
 ((((IDEATE WORD X4) (OUT-OF X4 *SIZE)) ((OUT-OF X1 X4)) STOP )
(TALK)
((IDEATE SQUARE X1)(IDEATE CIRCLE X2))
((OUT-OF X1 *SHAPE)(OUT-OF X2 *SHAPE))
((IDEATE RED X3)(IDEATE GREEN X4))
((OUT-OF X3 *COLOR)(OUT-OF X4 *COLOR))
(LISP SETQ X1 NIL)
((IDEATE SMALL X5)(IDEATE LARGE X1))
((OUT-OF X5 *SIZE)(OUT-OF X1 *SIZE))
NIL
(TALK)
((IDEATE TRIANGLE X1)(IDEATE BLUE X2)(IDEATE MEDIUM X3))
((OUT-OF X1 *SHAPE)(OUT-OF X2 *COLOR)(OUT-OF X3 *SIZE))
(LISP SETQ X1 NIL)
(LISP SETQ X2 NIL)
((IDEATE RIGHT-OF X1)(IDEATE ABOVE X2))
((OUT-OF RIGHT-OF *RA)(OUT-OF ABOVE *RA))
((OUT-OF LEFT-OF *RB)(OUT-OF BELOW *RB))
((IDEATE LEFT-OF X1)(IDEATE BELOW X2))
NIL
```

stood but was not able to speak. Also the child may have the potential to use a certain grammatical construction, but instead use other preferred modes of production. The final possibility is that the child may be resorting to non-linguistic strategies in language understanding. Bever (1970) has presented evidence that young children do not understand passives, but can still act out passives when they are not reversible. It seems the child can take advantage of the conceptual constraints between subject, verb, and object. The child's grammatical deficit only appears when asked to act out reversible passives. Similarly, Clark (1974) has shown that young children understand relational terms like in, on, and under by resorting to heuristic strategies. It is clear that we also have the ability to understand speech without knowing the syntax. For instance, when Tarzan utters food boy eat we know what he must mean. This is because we can take advantage of conceptual constraints among the words.

Bloom (1973) has also argued that the general belief that comprehension precedes production in a child is a misperception on the part of the adult observer. The study of Fraser, Bellugi, and Brown (1963) is often cited as showing comprehension precedes production. They found children had a higher probability of understanding a sentence (as manifested by pointing to an appropriate picture) than of spontaneously producing the sentence. However, there were difficulties of equating the measures of production and comprehension. Fernald (1970), using different scoring procedures, found no difference. Interestingly, Fraser et al. did find a strong correlation between which sentence forms could be understood and which could be produced. That is, sentence forms which were relatively easy to understand were relatively easy to produce. It is hard to understand this correlation except in terms of a common base for comprehension and production.

The SPEAK Program

SPEAK starts with a HAM network of propositions tagged as to-be-spoken and a topic of the sentence. The topic of the sentence will correspond to the first meaning-bearing element in the START network. SPEAK searches through its START network looking for some path that will express a to-be-spoken proposition attached to the topic and which expresses the topic as the first element. It determines whether a path accomplishes this by evaluating the actions associated with a path and determining if they created a structure that appropriately matches the to-be-spoken structure. When it finds such a path it uses it for generation.

Generation is accomplished by evaluating the conditions along the path. If a condition involves a push to an embedded network SPEAK is recursively called to speak some sub-phrase expressing a proposition attached to the main proposition. The arguments for a recursive call of PUSH are the embedded network and the node that connects the main proposition and the embedded proposition. If the condition does not involve a PUSH it will contain a set of memory commands specifying that some features be true of a word. It will use these features to determine what the word is. The word so determined will be spoken.

As an example, consider how SPEAK would generate a sentence corresponding to the HAM structure in Figure 6 using GRAMMAR2, the English-like grammar in Figure 5. Figure 6 contains a set of propositions about three objects denoted by the nodes G246, G195, and G182. Of node G246 it is asserted that it is a triangle, and that G195 is right of it. Of G195 it is asserted that it is a square and that it is above G182. Of G182 it is asserted that it is square, small, and red. Figure 7 illustrates the generation of this sentence from GRAMMAR2. LAS enters the START network intent on producing some utterance about G195. Thus, the topic is G195 (it could have been G246 or G182). The first path through the network involves predicating an adjective of G195, but there is nothing in the adjective class predicated of G195. The second path through the START network corresponds to something LAS can say about G195 -- it is above G182. Therefore, LAS plans to say this as its main proposition. First, it must find some noun phrase to express G195. The substructure under G195 in Figure 8 reflects the construction of this subnetwork. The NP network is called which prints *the* and calls NP1 which retrieves *square* and calls CLAUSE which prints *that*, *is*, and *right-of* and which recursively calls NP to print *the square*. Similarly, recursive calls are made on the NP1 network to express G182 as *the small red square*.

The actual sentence generated is dependent on choice of topic for the START network. Given the same to-be-spoken HAM network, but the topic *G246*, SPEAK generated *A triangle is left-of a square that is above a small red square*. Given the topic *G182* it generated *A red square that is below a square that is right-of a triangle is small*. Note how the choice of the relation words *left-of* vs. *right-of* and of *above* vs. *below* is dependent on choice of topic.

It is interesting to inquire what is the linguistic power of LAS as a speaker. Clearly it can generate any context-free language since its transition networks correspond, in structure, to a context-free grammar. However, it turns out that LAS has certain context-sensitive aspects because its productions are constrained by the requirement that they express some well-formed HAM conceptual structure. Consider two problems that Chomsky (1957) regarded as not handled well by context-free grammars: The first is agreement of number between a subject NP and verb. This is hard to arrange in a context-free grammar because the NP is already built by the time the choice of verb number must be made. The solution is trivial in LAS--when both the NP and verb are spoken their number is determined by inspection of whatever concept in the to-be-spoken structure underlies the subject. The other Chomsky example involves the identity of solutional restrictions for active and passive sentences. This is also achieved automatically in LAS, since the restrictions in both cases are regarded simply as reflections of restrictions in the semantic structure from which both sentences are spoken.

While LAS can handle those features of natural language suggestive of context-sensitive rules, it cannot handle examples like languages of the form $a^n b^n c^n$ which require context-sensitive grammars. It is interesting, however, that it is hard to find natural language sentences of this structure. The best I can come up with are respectively-type sentences, e.g., *John and Bill hit and kissed Jane and Mary, respectively*. This sentence is of questionable acceptabil

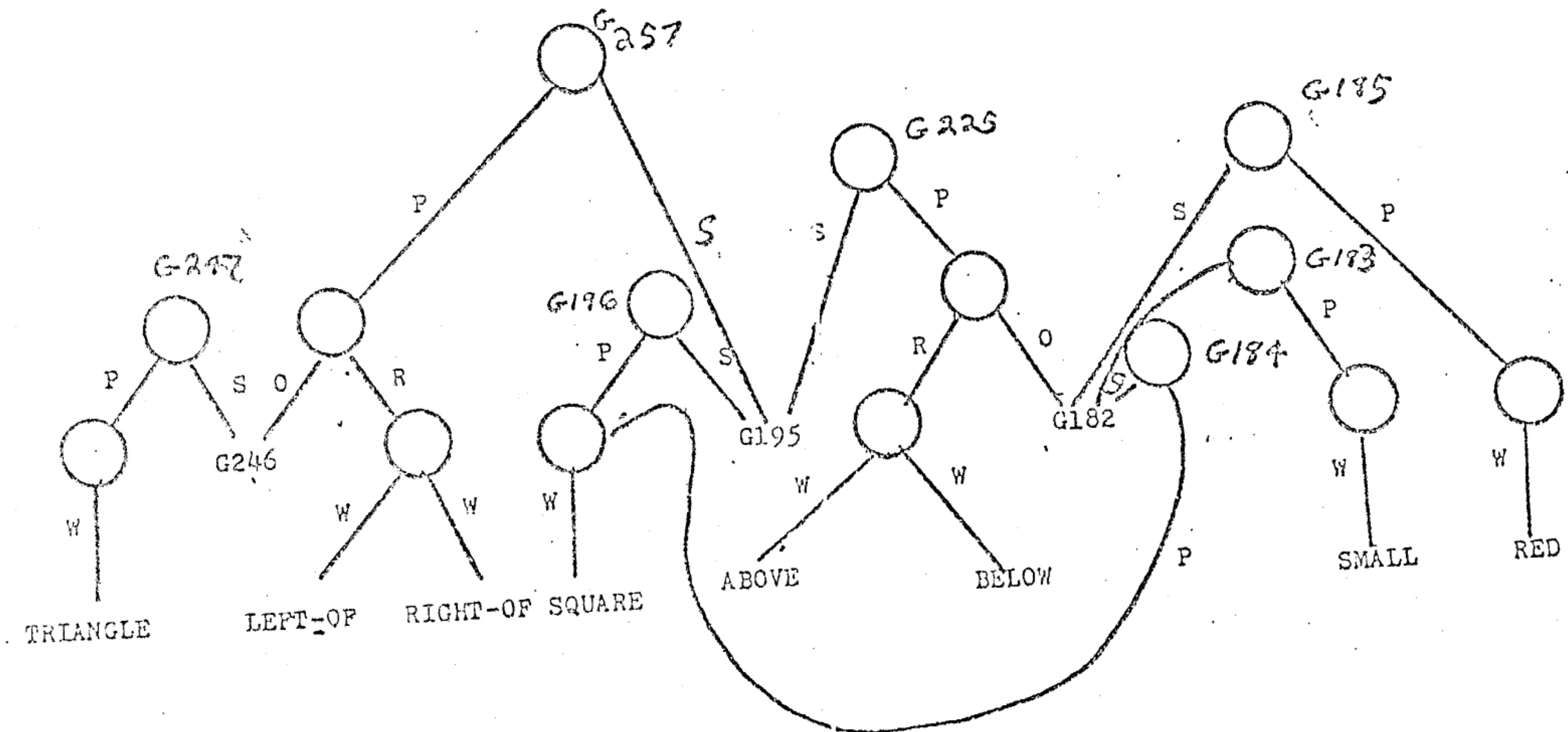


Figure 6. The to-be-spoken HAM network for the SPEAK program.

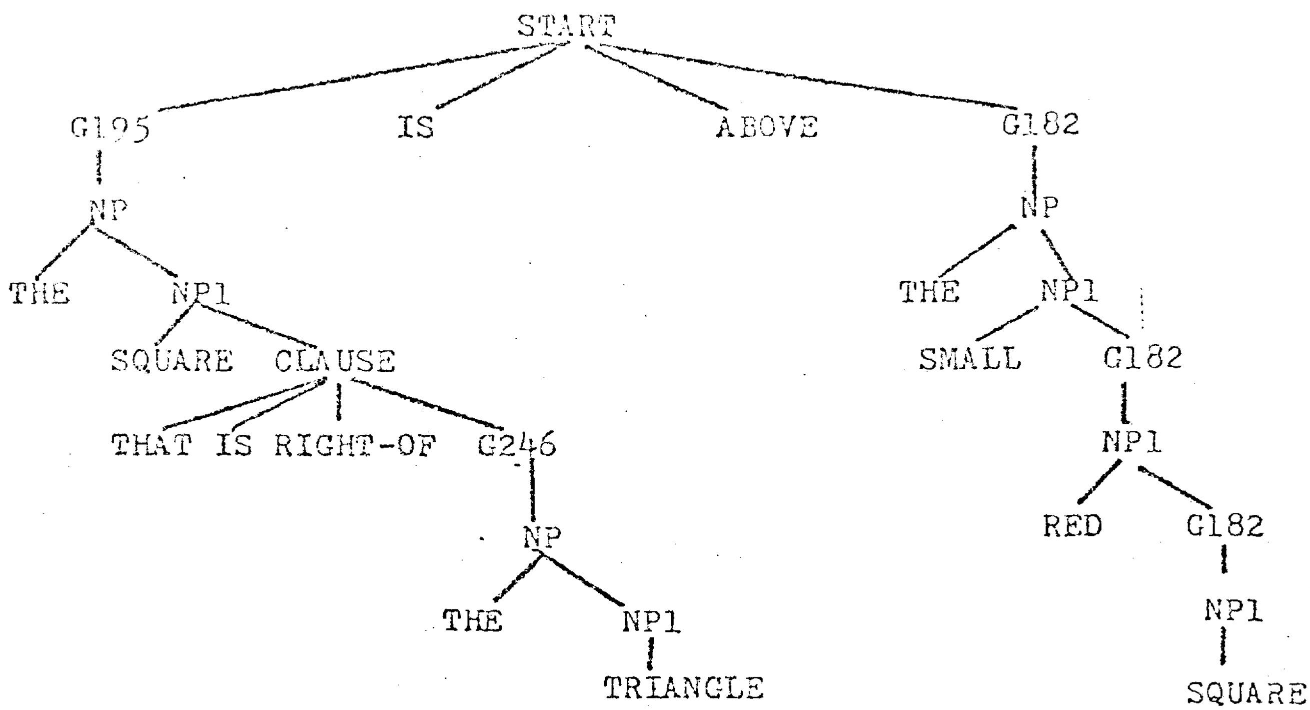


Figure 7. A tree structure showing the network calls and word output. These networks were called in generating a sentence about G195 which expressed the information contained in Figure 6.

The UNDERSTAND Program

The search in SPEAK for a grammatical realization of the conceptual structure was limited to search through a single network at a time. Search terminated when a path was found which would express part of the to-be-spoken HAM structure. Because search is limited to a single parsing network the control structure was simply required to execute a depth-first search through a finite network. In the UNDERSTAND program it is necessary, when one path through a network fails, to consider the possibility that the failure may be in a parsing of a subnetwork called on that path. Therefore, it is possible to have to back into a network a second time to attempt a different parsing. For this reason the control structure of the UNDERSTAND program is more complicated. The UNDERSTAND program and its control structure were written by Carol Hafner, a computer science student at Michigan.

Perhaps an English example would be useful to motivate the need for a complex control structure. Compare the two sentences The Democratic party hopes to win in '76 with The Democratic party hopes are high for '76. A main parsing network would call a noun phrase network to identify the first noun phrase. Suppose UNDERSTAND identified The Democratic party. Later elements in the second sentence would indicate that this choice was wrong. Therefore, the main network would have to re-enter the noun phrase network and attempt a different parsing to retrieve The Democratic party hopes. When UNDERSTAND re-entered the noun-phrase network to retrieve this parsing it must remember which parsings it tried the first time so that it does not retrieve the same old parsing. The complexities of this control structure are described in a more complete report (Anderson, 1975). Here I will just overview the general structure of the program. The program tries to find some path through the START network which will result in a complete parsing of the sentence. It evaluates the acceptability of a particular path by evaluating the conditions associated with that path. A condition may require that certain features be true of words in the sentence. This is determined by checking memory. Alternatively, a condition can require a push to an embedded network. This network must parse some subphrase of the sentence. When LAS finds an acceptable path through a network it will collect the actions along that path to create a temporary memory structure to represent the meaning of the phrase that LAS has parsed. This, for instance, given the sentence, The square that is right-of the triangle is above the small red square, LAS would parse it in the form illustrated for Figure 7, retrieving the HAM structure in Figure 6. That is, in LAS. 1, understanding really is simply generation put in reverse. This is the first displayed example of a reversible augmented transition network. Simmons (1973) comes closest with two different networks, one for generation and one for analysis.

It is also of interest to consider the power of LAS as an acceptor of languages. It is clear that LAS as presently constituted can accept exactly the context-free languages. This is because, unlike Woods' (1970) system, actions on arcs cannot influence the results of conditions on arcs, and therefore, play no role in determining whether a string is accepted or not. However, what is interesting is that LAS's behavior as an language understander is relatively little affected by its limitations on grammatical powers. Consider the following example of where it might seem that LAS would need a context-sensitive grammar: In English noun phrases, it seems we can have an arbitrary number of adjectives.

This led to the rule in GRAMMAR2 where NP1 could recursively call itself each time accepting another adjective. There is nothing in this rule to prevent it from accepting phrases like <u>the small big square</u> or other ungrammatical phrases. However, in practice this does not lead LAS into any difficulties because it would never be presented with such a sentence due to the constraints on what a speaker may properly say to LAS.

General Conditions for Language Acquisition

Having now reviewed how LAS. 1 understands and produces sentences, I will present the three aspects of the induction program: BRACKET, SPEAKTEST, and GENERALIZE. Before doing so, it is wise to briefly state the conditions under which LAS learns a language. It is assumed that LAS. 1 already has concepts attached to the words of the language. That is, lexicalization is complete. The task of LAS. 1 is to learn the grammar of the language--that is, how to go from a string of words to a representation of their combined meaning. Because LAS. 1 is not concerned with learning meanings, it cannot be a very realistic model for second language learning where many concepts can transfer from the first to the second language. I will propose extensions of LAS. 1 concerned with learning word meanings.

Another feature of LAS. 1 is that it works in a particularly restricted semantic domain. It is presented with pictures indicating relations and properties of two-dimensional geometric objects. These pictures are actually encoded into the HAM propositional network representation. Along with these pictures LAS is presented sentences describing the picture and an indication of that aspect in the picture which corresponds to the main proposition of the sentence. From this information input, a network grammar is constructed. The semantic domain may be very simple, but the goal is to be able to learn any natural or natural-like language which may describe that domain.

The BRACKET Program

A major aspect of the LAS project is the BRACKET program. This is an algori for taking a sentence of an arbitrary language and a HAM conceptual structure and producing a bracketing of the sentence that indicates its surface structure. This surface structure prescribes the hierarchy of networks required to parse the sentence. For BRACKET to succeed, four conditions must be satisfied by the information input to it:

<u>Condition 1</u>. All content words in the sentence correspond to elements in the conceptual structure. This amounts to the claim that the teacher is able to direct the learner to conceptualize the information in his sentence. It does not matte to the BRACKET algorithm whether there is more information in the conceptual structure than in the sentence.

<u>Condition 2</u>. The content words in the sentence are connected to the elements in the conceptual structure. Psychologically, this amounts to the claim that lexicalization is complete. That is, the learner knows the meanings of the wor

<u>Condition 3</u>. The surface structure interconnecting the content words is isomorphic in its connectivity to a language-free prototype structure.

Condition 4. The main proposition in conceptual structure is indicated.

Conditions 3 and 4 require considerable exposition. To explain Condition 3 I will first assume that the prototype structure is just the HAM conceptual structure. Later I will explain why something slightly different is required.

Consider Panel (a) of Figure 8 which illustrates the HAM structure for the series of propositions in the English sentence *The red square is above the small circle*. Panel (b) illustrates a graph deformation of that structure giving the surface structure of the sentence. Note how elements within the same noun phrase are appropriately assigned to the same subtree. Note that the prototype structure is not specific with respect to which links are above which others and which are right of which others. Although the HAM structure in Panel (a) is set forth in a particular spatial array, the choice is arbitrary. In contrast, the surface structure of a sentence does specify the spatial relation of links. It seems reasonable that all natural languages have as their semantics the same order-free prototype network. They differ from one another in (a) the spatial ordering their surface structure assigns to the network and (b) the insertion of non-meaning-bearing morphemes into the sentence. However, the surface structure of all natural languages is derived from the same graph patterns. Panel (c) of Figure 8 shows how the prototype structure of Panel (a) can provide the surface structure for a sentence of the artificial GRAMMAR1. All the sentences of GRAMMAR1 preserve the connectivity of the underlying HAM structure. By this criterion, at least, GRAMMAR1 could be a natural language.

However, certain conceivable languages would have surface structures which could not be deformations of the underlying structure. Panel (d) illustrates such a hypothetical language with the same syntactic structure as English, but with different rules of semantic interpretation. In this language the adjective phrase preceding the object noun modifies the subject noun. As Panel (d) illustrates, there is no deformation of the prototype structure in Panel (a) to achieve a surface structure for the sentences in the language. No matter how it is attempted some branches must cross.

LAS will use the connectivity of the prototype network to infer what the connectivity of the surface structure of the sentence must be. The network does not specify the right-left ordering of the branches or the above-below ordering. The right-left ordering can be inferred simply from the ordering of the words in the sentence. However, to specify the above-below ordering, BRACKET needs one further piece of information. Figure 9 illustrates an alternate surface structure that could have been assigned to the string in Figure 8 (c). It might be translated into English syntax as *Circular is the small thing that is below the red square*. Clearly, as these two structures illustrate, the HAM network and the sentences are not enough to specify the hierarchical ordering of subtrees in the surface structure. The difference between the sentences in Figure 8 (c) and 9 is the choice of which proposition is principal and which is subordinate. If BRACKET is also given information as to the main proposition it can then unambigiously retrieve the sentence's surface structure. The assumption that BRACKET is given the main proposition amounts, psychologically, to the claim that the teacher can direct the learner's attention to what is being asserted in the sentence. Thus, in Panel (c), the teacher would direct the learner to the picture of a red triangle above a small circle. He would both have to assume that the learner properly conceptualized the picture and that he also realized the aboveness relation was what was being asserted in the picture.

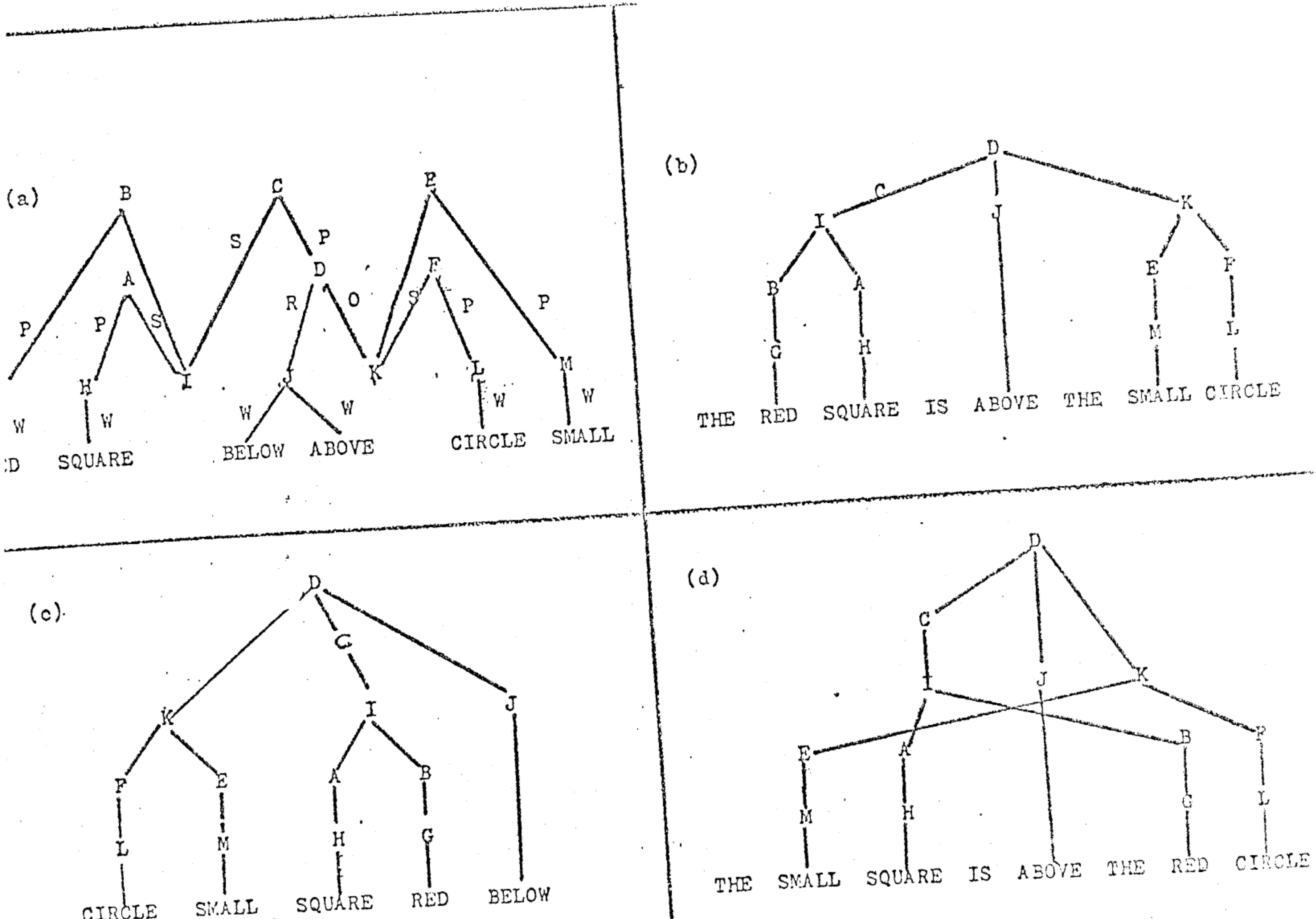


Figure 8. The surface structures of the sentences in (b) and (c) are graph deformations of the HAM structure in (a). Panel (d) [illegible] to deform (a) into a

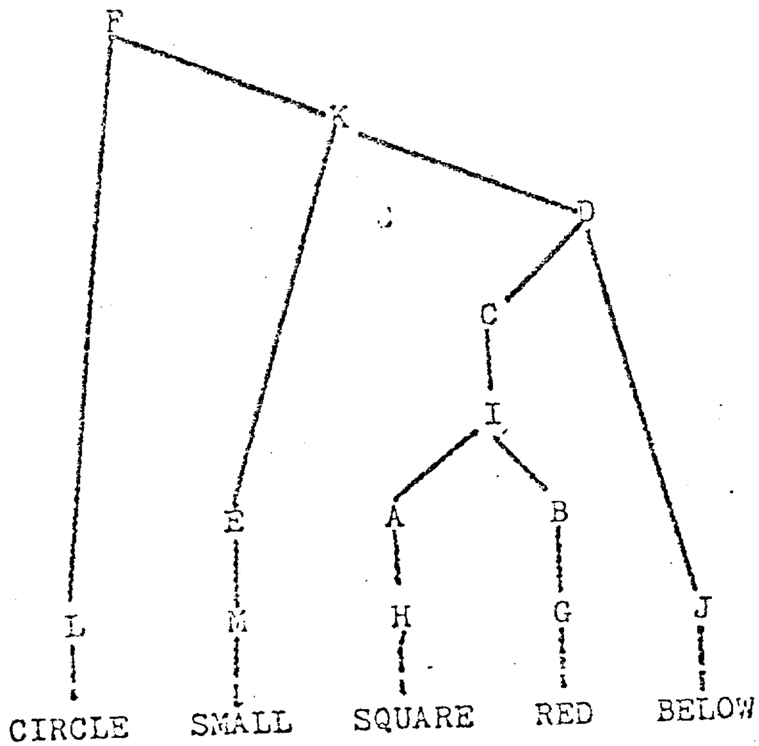


Figure 9. Alternate surface structure for the sentence in Figure 9c.

More on the Graph Deformation Condition

I think that the graph deformation condition has something of the status of a universal property of language. However, to make this claim viable it is clear that something other than the HAM network will have to be adopted as the prototype structure. HAM's binary branching works well enough for the domain of discourse that I have been interested in so far, but it will not generalize to sentences that have verbs that take more than two noun phrase arguments. Figure 10a shows how HAM would represent the sentence John opened the door with a key. This is decomposed into a set of sub-propositions--John turned the key which caused the door to be opened. Because of the binary structure certain elements are grouped together. In particular, John and key are closer together and door and open are closer together. If Figure 10a were the prototype, LAS could not bracket a sentence which alternated words from the two subgroups. For instance, there is no deformation of the structure in (a) that would provide a bracketing for John opened with a key the door. Branches of the HAM structure would have to cross. This English sentence and other English sentences which violate the deformation condition for Figure 10a have all a semi-unacceptable ring to them. However, this is almost certainly a peculiarity of English. Other languages permit free ordering of their noun phrases. What is needed for a prototype structure is something like the case representation in Figure 10b where all arguments are equally accessible from the main proposition node. The problem posed by the verb open is one posed by any verb which takes more than two noun phrase arguments. HAM's representation rules out certain sequences of the verb and its arguments while it is likely that all sequences can be found in some natural language. There are two ways to deal with this dilemma. One could resort to a memory representation like (b). However, there are a number of significant considerations that motivate the HAM representation in panel (a). Moreover, representations like (b) finesse one of the most interesting questions in language acquisition--how we learn the case structure of complex verbs. To address this question we need a representation that decomposes multi-argument verbs into a representation like (a) which exposes the semantic function of the case arguments. Learning the role of the verb open in the language then involves learning how to assign its noun phrase arguments to a structure like (a). I will sketch a system to do this in the proposal section.

If we keep the HAM representations then some changes are required in BRACKET graph deformation condition. What is characteristic of multi-argument verbs in HAM is that the arguments are interconnected by causal relations as in (a). Thus, BRACKET should be made to treat all the terminal arguments in such causal structures as defining a single level of nodes in a graph structure all connected to a single root node. That is, BRACKET can treat a HAM structure such as (a) if it were (b) for purposes of utilizing the graph deformation condition. In fact, BRACKET already does this in the current implementation.

The Details of BRACKET's Output

So far, only a description of how one would retrieve the surface structure connecting the content words of the sentence has been given. Suppose BRACKET were given A triangle is left-of a square that is above a small red square. A bracketing structure must be imposed on this sentence which will

(a)

(b)

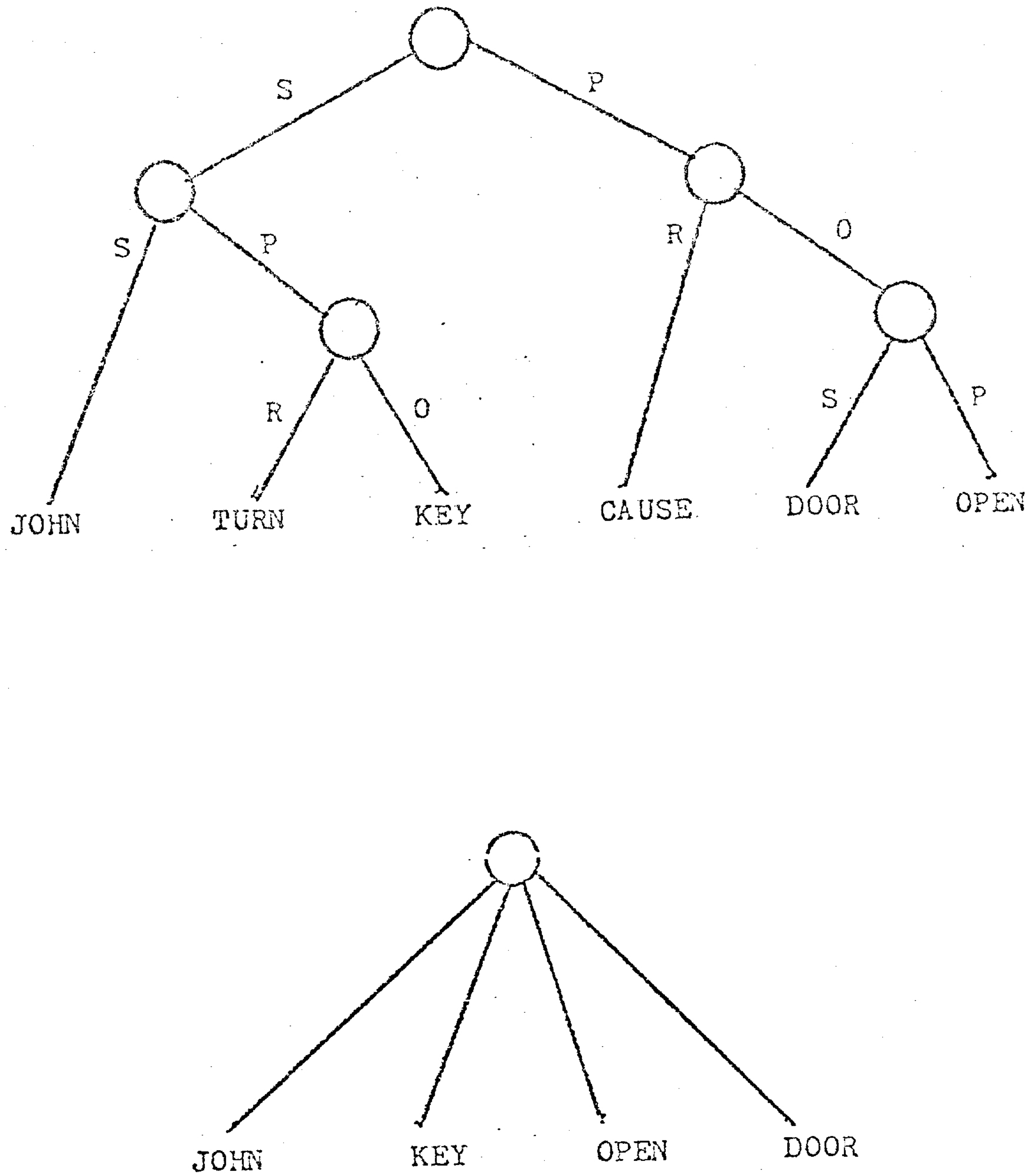


Figure 10. Alternative prototype structures for the sentence _John opened the door with a key_. The HAM structure in (a) introduces too many distinctions.

also include the function words. Given this sentence and the conceptual structure in Figure 6, BRACKET returned (G257 (G246 G247 a triangle) is left-of (G195 G196 a square (G195 G225 that is above (G182 G183 a small (G182 G185 red (G182 G184 square)))))). The main proposition is G257 which is given as the first term in the bracketing. The first bracketed sub-expression describes the subject noun phrase. The first element in the sub-expression G246 is the node that links the embedded proposition G247 to the main proposition G257. The first two words of the sentence *A triangle* are placed in this bracketed sub-expression. The next two words *is left-of* are in main bracketing. There are no embedded propositions corresponding to these two. The remainder of the output of BRACKET corresponds to a description of the element G195. The first embedded proposition G196 asserts this object is a square and the second proposition, G225, asserts that G195 is above G182. Note that the G225 proposition is embedded as a sub-expression within the G196 proposition. The last element in the G225 proposition is (G182 G183 a small (G182 G185 red (G182 G184 square))). This expression has in it three propositions G183, G185, G184 about G182.

The above example illustrates the output of BRACKET. Abstractly, the output of BRACKET may be specified by the following three rewrite rules:

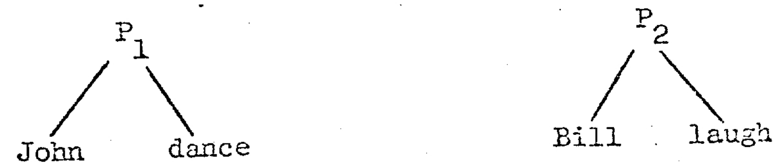
1. S → proposition element *
2. element → word
3. element → (topic S)

That is, each bracketed output is a proposition node followed by a sequence of elements (rule 1). These elements are either rewritten as words (rule 2) or bracketed subexpressions (rule 3). A bracketed subexpression begins with a topic node which indicates the connection between the embedded and embedding propositions. The elements within an expression are either non-meaning bearing words or elements corresponding to subject, predicate, relation and object in the proposition. Note that BRACKET induces a correspondence between a level of bracketing and a single proposition. Each level of bracketing will also correspond to a new network in LAS's grammar. Because of the modularity of HAM propositions, a modularity is achieved for the grammatical networks. When a number of embedded propositions are attached to the same node, they are embedded within one another in a right-branching manner.

The insertion of non-function words into the bracketing is a troublesome problem because there is no semantic features to indicate where they belong. Consider the first word *a* in the example sentence above in Figure 6. It could have been placed in the top level of bracketing or in the subexpression containing triangle. Currently, all the function words to the right of a content word are placed in the same level as the content word. The bracketing is closed immediately after this content word. Therefore, *is* is not placed in the noun-phrase bracketing. This heuristic seems to work more often than not. However, there clearly are cases where it will not work. Consider the sentence *The boy who Jane spoke to was deaf*. The current BRACKET program would return this as ((The boy (who Jane spoke)) to was deaf). That is, it would not identify *to* as in the relative clause. Similarly, non-meaning-bearing suffixes like gender would not be retrieved as part of the noun by this heuristic. However, there is a strong cue to make bracketing appropriate in these cases. There tends to be a pause after morphemes like *to*. Perhaps such pause structures could be called upon to help the BRACKET program decide how to insert the non-meaning-bearing morphemes into the bracketing.

Non-meaning-bearing morphemes pose further problems besides bracketing. Consider a sequence of such morphemes in a noun phrase. That sequence could have its own grammar that, in principle, might constitute an arbitrary recursive language. The sentence's semantic referent could provide no cues at all as to the structure of that language. Therefore, we would be back to the same impossible language induction task that we characterized in the introduction. Hence, it is comforting to observe that the structure of these strings of non-meaning-bearing morphemes tends to be very simple. There are not many examples of these strings being longer than a single word. Thus, it seems that the languages constituted by these non-meaning-bearing strings are nothing more than very simple finite cardinality languages which pose, in themselves, no serious induction problems. The various stretches of non-meaning-bearing morphemes in a sentence could also have complex interdependencies thereby posing serious induction problems. Again it does not seem to be the case that these dependencies exist. So once again we find that the structure of natural language is simple just at those points where it would have to be for a LAS-like induction program to work.

In concluding this section I should point out one example sentence which BRACKET cannot currently handle. They are respectively sentences like John and Bill danced and laughed respectively. The problem will such a sentence is that underlying it is the following prototype structure:



Thus, John and dance are close together and so are Bill and laugh. However, the sentence intersperses these elements just in the way that makes bracketing impossible. There are probably other examples like this, but I cannot think of them. Fortunately, this is not an utterance that appears early in child speech nor is a particularly simple one for adults. Of all the grammatical constructions, the respectively construction is the one that most suggests the need to have transformational rules in the grammar.

SPEAKTEST

The function of SPEAKTEST is to test whether its grammar is capable of generating a sentence and, if it is not, appropriately modify the grammar so that it can. SPEAKTEST is called after BRACKET is complete. It receives from BRACKET a HAM conceptual structure, a bracketed sentence, the main proposition and the topic of the sentence. As in the SPEAK program SPEAKTEST attempts to find some path through its network which will express a proposition attached to the topic. If it succeeds no modifications are made to the network. If it cannot, a new path is built through the network to incorporate the sentence.

The best way to understand the operation of SPEAKTEST is to watch it go through one example. The target language it was given to learn is illustrated in Table 4. This is a very simple language, basically GRAMMAR1 of Table 1. It has a smaller vocabulary to make it more tractable. The reason for choosing this language is that it is of just sufficient complexity to illustrate LAS's acquisition mechanisms. In addition, LAS has learned GRAMMAR2, also given in Table 1.

Figure 11 illustrates LAS's handling of the first two sentences that come in. The first sentence is _Square triangle above_. This sentence is returned by BRACKET as (G174 (G115 G116 square) (G148 G149 triangle) above). _G174_ refers to the main proposition given as an argument to LEARNMORE. Since this is LAS's first sentence of the language the START network will, of course, completely fail to parse the sentence. It has no grammar yet. Therefore, it induces the top-level START network in Figure 11. A listing of the exact arc information induced is given below the graphical illustration in Figure 11. Since the first two elements after _G174_ in the bracketed sentence are themselves bracketed, the first two arcs in the network will be pushes to subnetworks. The third arc contains a condition on the word _above_. The restriction made is that it be a member of the word class _A199_. This class was created for this sentence and only contains the word _above_ at this point. Having now constructed a path through the START network, SPEAKTEST checks the subnetworks in that path to see whether they can handle the bracketed subexpressions in the sentence. This is accomplished by a recursive call to SPEAKTEST. For the first phrase, SPEAKTEST is called, taking as arguments the network A195, the phrase (G116 square) and the topic G115. In network A195 the word class _A211_ is created to contain _square_, and in network _A197_ the word class _A221_ contains _triangle_. These two subnetworks should be the same in a final grammar but LAS is not prepared to risk such a generalization at this point.

Note in this example how the bracketing provided by BRACKET completely specified the embedding of networks. The sentence provided by BRACKET was (G174 (G115 G116 square) (G148 G149 triangle) above). The first element _G174_ was the main proposition. The second element (G115 G116 square) was a bracketed subexpression indicating a subnetwork should be created. Similarly, the third expression indicated a subnetwork. The last element _above_ was a single word and so could be handled by a memory condition in the main network.

The second sentence is _triangle square right-of_. This is transformed by BRACKET to (G315 (G246 G247 triangle) (G283 G284 square) right-of). Because of the narrow one-member word classes this sentence cannot be handled by the current grammar. However, SPEAKTEST does not add new network arcs to handle the sentence. Rather, it expands word class _A199_ to include _right-of_, word class _A211_ to include _triangle_, and word class _A221_ to include _square_. The grammar is now at such a stage that LAS could speak or understand the sentences _triangle square above_ or _square square right-of_ and other sentences which it had not studied. Thus, already the first generalizations have been made. LAS can produce and understand novel sentences.

This illustrates the type of generalizations that are made within the SPEAKTEST program. For instance, consider the generalization that arose when SPEAKTEST decided to use the existing network structure to incorporate _triangle_,

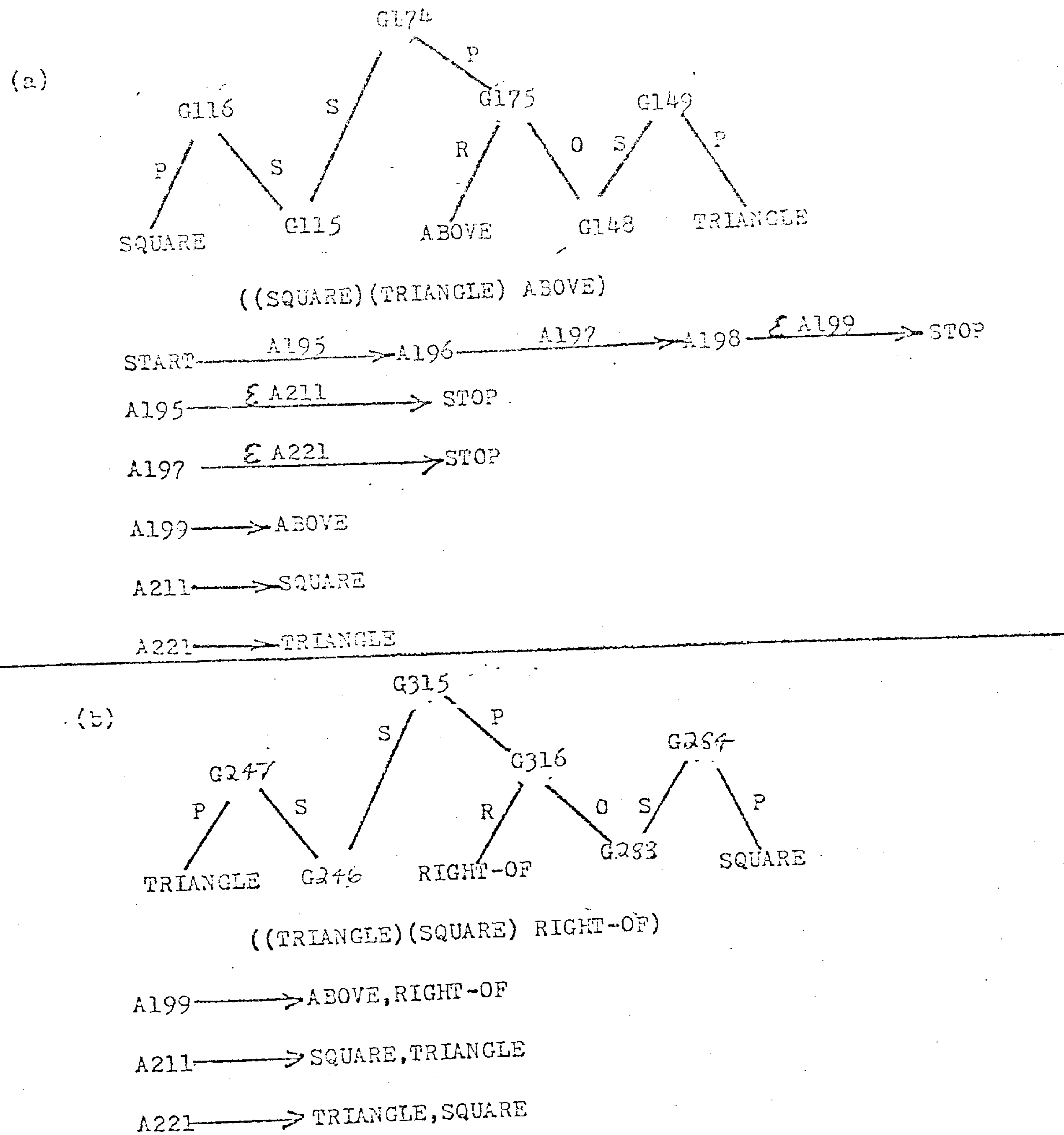


Figure 1|. IAS's treatment of the first two sentences in the induction sequence.

the first word of the second sentence. This involved (a) using the same subnetwork A195 that had been created for square and (b) expanding the word class A211 to include triangle. Both decisions rested on semantic criteria. The network A195 was created to analyze a description of a node attached to the main proposition by the relation S. Triangle was a description of the node G246 which is related by S to the main proposition. On the basis of this identity of semantic function, LAS assigns the parsing of triangle to the network A195 Within the A195 network the word class A211 contains words which are predicates of the subject node. Triangle has this semantic function and is therefore added to the word class.

In making these generalizations, SPEAKTEST is making a strong assumption about the nature of natural language. This assumption is stated as Condition 5:

Condition 5. Words or phrases with identical semantic functions at identical points in a network behave identically syntactically. This is the assumption of semantic-induced equivalence of syntax. It is another way in which semantic information facilitates grammar induction. It clearly need not be true of an arbitrary language. For instance, decisions made in the subject noun phrase might in theory condition syntactic decisions made in the object noun phrases. LAS, because of its heuristics in SPEAKTEST for generalization, would not be able to learn such a language.

Figure 12 illustrates LAS's network grammar after two more sentences have come in. Sentences 3 and 4 involve the relations below and left-of. LAS treats these as syntactic variants of above and right-of which differ in their assignment of their noun phrase arguments to the logical categories subject and object. Therefore, LAS creates an alternative branch through its START network to accommodate this possibility.

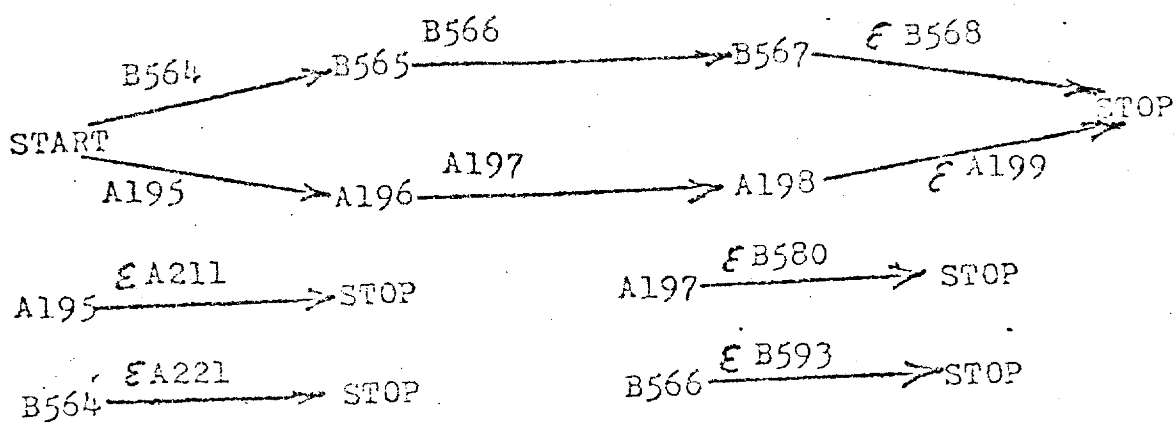
Figure 13 illustrates the course of LAS's learning. Altogether LAS will be presented 14 sentences. Subsequently, it will have to make three extra generalizations to capture the entire target language. Plotted on the abscissa is this learning history and along the ordinate we have the natural logarithm of the number of sentences which the grammar can handle. This is a finite language, unlike GRAMMAR2, and therefore the number of sentences in the language will always be finite. As can be seen from Figure 13, by the fourth sentence LAS's grammar is adequate to handle 16 sentences.

LAS's grammar after the next five sentences is illustrated in Figure 14. These are LAS's first encounters with two word noun phrases. All five sentences involve the relations right-of and above and therefore result in the elaboration of the A195 and A197 sub-networks. Consider the first sentence, square red triangle blue above, which is retrieved by BRACKET as (C329 (C270 C271 square (C270 C272 red)) (C303 C304 triangle (C303 C305 blue) above) C270). Consider the parsing of the first noun phrase. Note that the adjective (C270 C272 red) is embedded within the larger noun phrase. This is an example of the right embedding which BRACKET always imposes on a sentence. This will cause SPEAKTEST to create a push to an embedded network within its A195 subnetwork. As can be seen in Figure 14, the existing arc containing the A211 word class is kept to handle square. Two alternative arcs are added--one with a push to

Figure 12

LAS's grammar after studying:

1. SQUARE TRIANGLE ABOVE
2. TRIANGLE SQUARE RIGHT-OF
3. SQUARE TRIANGLE BELOW
4. TRIANGLE SQUARE LEFT-OF



A199 = above,right-of
A211 = square,triangle
A221 = square,triangle
B568 = below,left-of
B580 = square,triangle
B593 = square,triangle

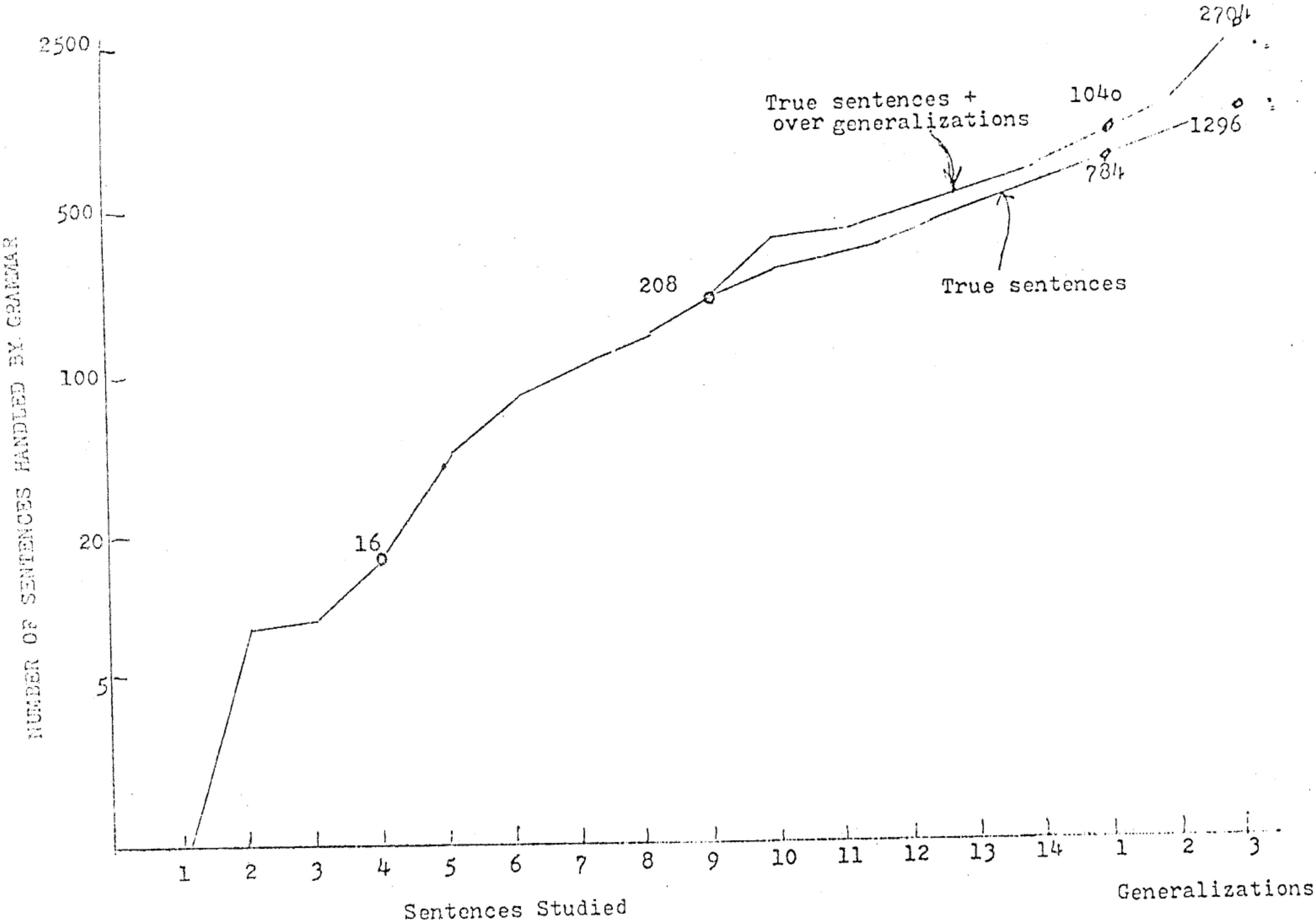
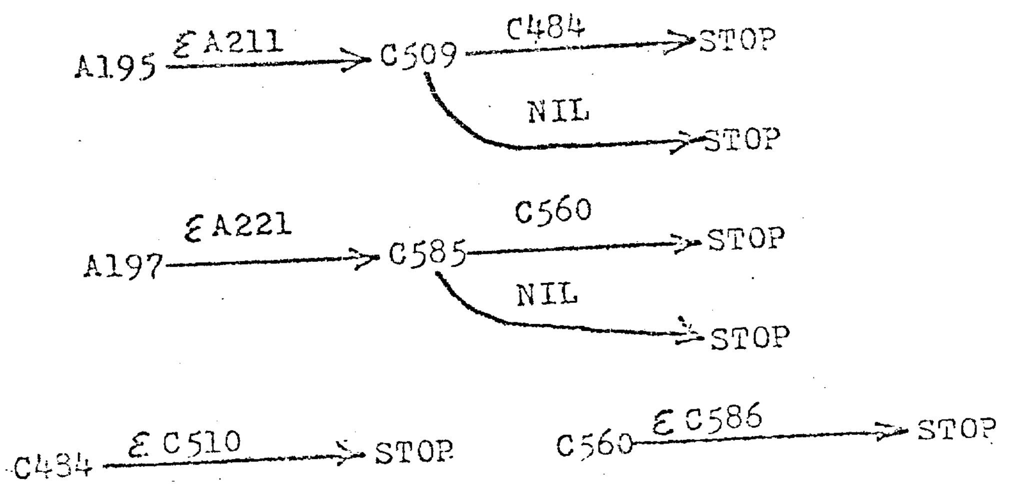


Figure 13. The growth of LAS's grammar with its learning history.

Figure 14

Additions to LAS's grammar after studying:

1. SQUARE RED TRIANGLE BLUE ABOVE
2. TRIANGLE LARGE SQUARE SMALL RIGHT-OF
3. TRIANGLE RED TRIANGLE RED ABOVE
4. SQUARE SMALL TRIANGLE RED RIGHT-OF
5. SQUARE BLUE TRIANGLE LARGE RIGHT-OF



C510 = small,blue,large,red
C586 = small,blue,large,red

the *C494* network and the other with a NIL transition. Within the *C494* network the word class *C510* is set up which initially only contains the word *red*.

This illustrates the principle of left generalization in LAS: Suppose a network contains a sequence of arcs A_1, A_2, ... A_m. Suppose further a phrase assigned to the network requires arcs X_1 ... X_m ... X_n to be successfully parsed. If arcs A_1, A_2, ... A_m have the same semantic function as required of arcs X_1, X_2, ... X_m, then the parsing of the first *m* elements in the phrase is assigned to the existing arcs A_1 ... A_m. After arc A_m two alternate paths are built. A NIL arc is added to permit the phrases that used to be parsed by A_1 ... A_m. Also arcs X_{m+1} ... X_n are added to handle the new phrase. LAS is making the generalization that any sequence of constituents parsable by A_1 ... A_m can be placed in front of any sequence of elements parsable by X_{m+1} ... X_n. Left generalization may be seen as an elaboration of semantics-induced equivalence of syntax (Condition 5).

Figure 15 illustrates a more conservative way that LAS might have made this generalization. Instead of network (a), it might have set up network (b). In network (b) a new word class *X* has been set up to record just those words which can be followed by an adjective. Networks (c) and (d) illustrate how left generalization can and does lead to overgeneralization in natural language. Suppose a child hears phrases like *The boy*, *A dog*, *the foot*, etc. He would set up a network that would accept any article followed by any noun. Suppose he then hears *The boys*.[1] This would be represented in LAS as *The* + *boy* + *'s*. Because of left-generalization LAS would construct the network illustrated in (d). In this network LAS has incorporated the generalization that *foots* is the pluralization of *foot*. This sort of morphemic generalization is, of course, a notorious overgeneralization in child language (e.g., Ervin, 1964). What is distinctive about such morphemic rules is that there are a number of alternatives and no semantic basis to choose between them. Because of its principle of semantics-induced equivalence of syntax, LAS will overgeneralize in those situations. Apparently, children are operating under a similar rule.

LAS needs to be endowed with a mechanism to allow it to recover from such overgeneralizations. Therefore, one of the future additions to LAS will have to be a RECOVER program. Consider how it would work with this pluralization example. Suppose LEARNMORE receives the sentence *The feet are above the triangle*. In attempting to analyze the sentence in SPEAKTEST, the plural *foots* will be generated but will mismatch the sentence. RECOVER has as its function to note such mismatches. Since it is possible that there are two alternate ways of expressing plurality, RECOVER cannot assume its grammar is wrong. Rather it will interrupt the information flow and check the acceptability of *The foots are above the triangle*. That is, RECOVER will explicitly seek negative information. Upon learning the expression is ungrammatical RECOVER will take *foot* out of the word class that is pluralized by 's.

---

1 To accomplish this I would have to put within LAS some mechanism that will segment words into their morphemes.