UNDERSTANDING SPEECH IN THE HEARSAY-II SYSTEM

Frederick Hayes-Roth
Rand Corporation

D. Jack Mostow
Carnegie-Mellon University

Mark S. Fox
Carnegie-Mellon University

1 OVERVIEW OF THE SPEECH UNDERSTANDING PROBLEM

A human speaker who wishes to express a particular thought translates his or her intention into words and then sound. The speech understanding problem consists of analyzing this sound, recognizing the spoken words, and inferring the speaker's intention. Uncertainty prevails at every stage during the transduction of a spoken utterance from a speech waveform to a word-for-word transcription and meaning interpretation. These uncertainties impede development of procedures to perform basic speech understanding functions, including: dividing the speech signal into relatively invariable acoustic segments; assigning a correct phonetic label to a segment; combining segments into a syllable; determining where one word ends and another begins in time; deciding which word was uttered in a given time interval; and interpreting the intention of a recognized sequence of words.

There are two basic causes of uncertainty in speech processing: signal variability and system limitations. Spoken language is characterized by extreme variability in the possible realizations of a speaker's intent to utter a particular thought, word, or phoneme. There are many different ways to express a thought in words, to pronounce (or mispronounce) a word, and to enunciate (or garble) a phoneme. This variability in the set of speech signals that can communicate the same message is exacerbated by varying environmental noise and distortions introduced by microphones used for converting speech input into electronic form.

The second basic type of uncertainty originates in the speech understanding system itself. The mechanisms for recognizing phonetic segments, syllables, and words are based on knowledge that is incomplete and errorful. Moreover, since each mechanism works on errorful input, its output is necessarily uncertain. In a successful speech understanding system, these mechanisms must cooperate in such a way that the errors they make cancel out rather than combine.

A speech understanding system can cope with variable input and imperfect recognition mechanisms by applying many diverse types of knowledge to the recognition problem, including: acoustic, phonetic, prosodic, syllabic, lexical, syntactic, semantic, pragmatic, and contextual. Each of these types of knowledge is insufficient by itself to solve the speech understanding problem, but in combination they can be used effectively to recognize spoken utterances.

Psychological experiments have shown that humans use all of these types of knowledge in recognizing speech, in particular the "higher-level" knowledge. Human performance in transcribing utterances decreases significantly when the utterances are inconsistent with one or more types of higher-level knowledge or when such knowledge is inapplicable. For example, error rates increase when sentences are heard out of context (pragmatically inconsistent), are nonsensical (semantically inconsistent), or are in a language unknown to the subject (lexically inconsistent) [15]. Furthermore, the state of the art performance in automatic "bottom-up" word recognition -- i.e., recognition of connected speech without the use of higher-level knowledge -- is substantially inferior to human performance [16]. Thus, incorporating this type of knowledge is essential for artificial speech understanding systems.

A fundamental problem in the design of a speech understanding system is organizing diverse sources of knowledge so they can be applied effectively to the problem of understanding an utterance. Each type of knowledge can most readily be applied when the utterance is represented in a particular form, and diverse types of knowledge induce varying representations of characteristics of the utterance. For example, a phonetic representation is convenient for applying knowledge about how words are pronounced but is a poor basis for the application of syntactic or semantic knowledge. The various levels of representation suggest a hierarchical problem-solving organization, in which each source of knowledge can be applied to data represented at one level in order to produce an interpretation of that data at an adjacent level. The importance of representing the utterance simultaneously in several forms further complicates the design of a speech understanding system.

There are many different but potentially useful problem solving methods for analyzing an utterance. Different methods reflect different ways of applying the same knowledge about speech. Of course, the most economical and efficient methods are most preferred. A method is called *strong* if it applies a relatively large amount of knowledge or requires relatively complete input data to solve a problem [13]. As the precision of its constraints is increased, a strong method simultaneously becomes more efficient and less generally applicable. A method is considered *weak* if it applies relatively little knowledge or places little constraint on its input data. Weak methods are more suitable for application to incomplete data than strong methods, but they tend both to be inefficient and to generate many extraneous results. Thus a speech understanding system may need an assortment of strong and weak problem solving methods, although they may be redundant in representing different ways of applying the same knowledge. The decision to include a particular method in a speech understanding system depends on its cost-effectiveness when combined with the other methods being used. An experimental speech understanding system should therefore be

designed to facilitate the exploration of different problem solving methods and the interactions between them. This chapter describes the application of syntactic and semantic knowledge in the HEARSAY-II speech understanding system at Carnegie-Mellon University and the exploration of different problem solving methods during two years of experimentation.

2 *ORGANIZATION OF KNOWLEDGE IN HEARSAY-II*

In the HEARSAY-II speech understanding system, different types of speech knowledge are encoded in *knowledge source modules* (KSs). KSs communicate with one another only by reading, evaluating, correlating, and modifying hypotheses about the utterance on a global data base called the *blackboard*. An hypothesis represents an interpretation of some temporal interval of the utterance at a particular *level of representation*, including acoustic, phonetic, syllabic, lexical, or phrasal. Hypotheses related implicatively -- for example, a sequence of hypotheses at one level supporting (providing evidence for) an hypothesis at a higher level -- are connected explicitly on the blackboard by directed arcs called *links*. The overall goal of the system is to translate the utterance from its initial acoustic representation to successively higher levels with the ultimate purpose of generating an hypothesis spanning the entire utterance that provides a plausible interpretation at the phrasal level (i.e., a parse).

The blackboard is organized uniformly along the dimensions of *time* and *level of representation*. An hypothesis that interprets a portion of the utterance at a particular level of representation -- e.g., postulates that the word "tell" was uttered in the time interval [10:30] (time measured in centiseconds past the beginning of the utterance) -- is associated with the appropriate level (lexical in this example) and has *begin* and *end times* 10 and 30. Since these terminal times are somewhat uncertain, they are represented with *ranges* indicating their degree of fuzziness. For example, "tell[10$\pm$2:30$\rightarrow$]" hypothesizes that the word "tell" starts sometime between 10-2=8 and 10+2=12 csec. following the beginning of the utterance and ends sometime at or after 30 csec. Alternative interpretations of the same interval at the same level are represented by different hypotheses. The alternative hypotheses can be considered to be distributed along a third nominal dimension of possibility or variety. Thus a second hypotheses "give[8$\pm$1:25$\pm$2]" could be represented simultaneously with competing hypothesis "tell[10$\pm$2:30]".

Processing of an utterance in HEARSAY-II exemplifies the hypothesize-and-test paradigm of problem-solving [12]. Control of processing is distributed among the various KS modules. Each KS module includes both a *precondition* and a *procedure* (KS for short). When the precondition detects a configuration of hypotheses to which the KS's knowledge can be applied, it *invokes* the KS procedure, i.e., schedules a blackboard-modifying operation by the KS. When this operation is performed, the resulting changes to the blackboard trigger the preconditions of other KSs and entail further activity (KS invocations). This data-directed propagation of activity can occur multidirectionally -- bottom-up (toward higher levels of representation), top-down (translation of hypotheses into lower-level representations for purposes of evaluation), or outward along the dimension of time (hypothesization based on surrounding context). For example, an earlier hypothesization of the word "tell" spanning the time interval [10:30] could lead to the generation of hypotheses that:

(1) a phrase of the form "Tell me about X" starts at time 10 and ends sometime after 30 (bottom-up recognition);
(2) the sequence of phonemes "T," "EH," and "L" occurs in the time interval [10:30] (top-down elaboration);
(3) a word in the category {me, us} starts at time 30 (lateral prediction).

Any newly generated hypotheses would be connected by links to the seminal hypothesis to indicate the implicative or evidentiary relationship between them.

In general, many inferences -- knowledge source invocations -- are potentially warranted by the configuration of blackboard hypotheses extant at any point during analysis of the utterance. If all possible inferences are made in an arbitrary "first come, first served" order, a combinatorial explosion of predominantly useless inferencing behavior results. Thus, some mechanism is needed to focus the attention of the system in promising directions [2]. A potential inference is considered promising to the extent that it is:

(1) *reliable* -- has a high probability of being correct;
(2) *necessary* -- will not duplicate an existing result; and
(3) *useful* -- will contribute toward the overall goal of recognizing the utterance.

Two basic methods are used to solve this *focus of attention* problem in HEARSAY-II [2]. The first constrains the hypothesization process by defining thresholds of minimum acceptable reliability for inferences. The reliability of a proposed inference depends both on the plausibility of the evidence on which it is based and on the strength of the inference itself, i.e., the conditional probability that the inference is correct given valid supporting evidence. The plausibility of an hypothesis is represented numerically by its *validity*, an integer between 100 (maximally plausible) and -100 (maximally implausible). Similarly, the conditional strength of an inference is represented by an *implication* value, ranging from 100 (maximally confirming evidence) to -100 (maximally disconfirming evidence). The validity of an hypothesis is a function of the validity of the hypotheses directly supporting it via implicative links and the implicative strengths associated with those links. Changes in validity ratings reflecting creation and modification of hypotheses and links are propagated automatically throughout the blackboard by a *rating policy module* called RPOL [9]. RPOL insures that the validity of an hypothesis is always the best current estimate of its plausibility. Other information pertaining to hypotheses can be specified via attributes created or modified by KSs. Particular attributes are defined for hypotheses at various levels. Any KSs that know about a particular attribute can access or change its value.

Given these measures of hypothesis validity and implicative strength, the hypothesization process can be constrained by defining minimum thresholds on:

(1) the validity of the hypotheses on which an inference is based, i.e., the plausibility of the data adduced as evidence for the inference;
(2) the implicative strength of the inference, i.e., the reliability of the permanent knowledge on which the inference is based; and
(3) the expected validity of the hypothesis produced by the inference, i.e., the estimated plausibility of the result.

All three of these thresholding methods are used in HEARSAY-II to reduce the number of hypotheses created. Thresholds can be defined with values local to a particular time interval and level of representation in response to variations in data reliability in different regions of the blackboard. Moreover, thresholds can be lowered or raised to increase or reduce the amount of hypothesization permitted. Any relaxation of a threshold constraint necessitates the reconsideration of previously rejected inferences. Thus KSs must be sensitive to threshold changes as well as to data changes. Threshold values are determined by a *focussing strategy* module in a manner consistent with the desired search policy (e.g., bottom-up, breadth-first, best-first).

The thresholding mechanism is absolute in nature: it decides whether or not to permit a given inference. In contrast, the second mechanism, *priority scheduling*, is relative: it decides what priority to assign to the execution of a proposed inference (knowledge source invocation). Inferences with highest priority are performed first; lower-priority inferences are deferred (possibly forever) until no higher-priority inferences remain to be executed. The priority of an inference is a function of its reliability, necessity, and utility. These can change according to developments on the blackboard; e.g., a pending inference is obviated if the result it would have produced is achieved in the interim by other means. Therefore the priority of a pending scheduled inference is reevaluated periodically to reflect the best estimate of the inference's promise given the current blackboard configuration. Scheduling priorities are determined by a scheduling policy based on the factors described above.

3 SYNTACTIC AND SEMANTIC KNOWLEDGE

The function of syntactic and semantic knowledge in a speech understanding system is to identify correctly the spoken utterance in a combinatorially large *search space* of sequences of hypothesized words. This search space can be graphically represented by a chart of the blackboard at the word level as in Figure 3.1. In the chart, the horizontal dimension is time, measured in csec. elapsed since the beginning

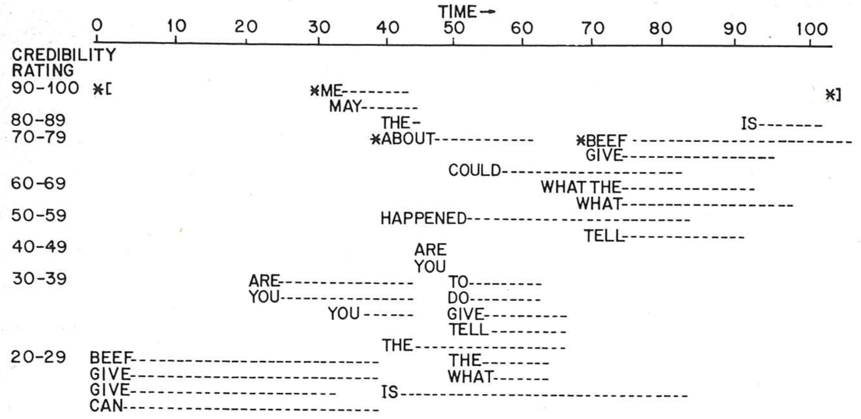

Figure 3.1: Words hypothesized bottom-up in response to utterance "Tell me about beef." "*" marks correct hypotheses "[" and "]" denote hypothesized beginning and ending of utterance.

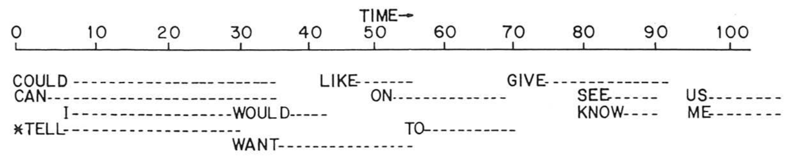

Figure 3.2: Words predicted by SASS on the basis of the hypotheses shown in Figure 3.1.

of the utterance. The vertical dimension is hypothesis validity rating. Since some of the uttered words may not be hypothesized by lower-level KSs, this search space is not guaranteed to contain the correct solution. Hence the system must be able to hypothesize syntactically and semantically plausible words not detected by other methods. This capability for top-down *prediction* can be thought of as expanding the search space.

Since bottom-up word recognition is imperfect in HEARSAY-II, the data (word hypotheses) contain a large amount of noise. Typically (for a 1,000-word vocabulary) only 80% of the uttered words are hypothesized bottom-up and, for each correctly recognized word, an average of four incorrect words are hypothesized in the same time interval with a higher validity rating. Knowledge of syntactic and semantic constraints on what constitutes a plausible word sequence can be used to filter out this "noise."

The KS responsible for syntactic and semantic operations in HEARSAY-II is called SASS. The basic task of SASS is to parse the utterance. SASS must find plausible word sequences spanning the utterance, e.g., [TELL ME ABOUT BEEF] in Figure 3.1. In the process, SASS must store and use partial parse information; for example, the plausible, grammatical subsequence ME ABOUT BEEF should be recognized. SASS must select the most plausible complete parse from those it finds and, finally, it must interpret the recognized utterance, which in this case is a request for information about "beef." If the utterance is only partially recognized, the recognized fragments should be interpreted [7].

The concept of a word sequence in the domain of speech relies on a generalized definition of word *adjacency* not necessary in simpler problems, such as understanding teletyped input. SASS must tolerate fuzzy and inaccurate word boundaries. Figure 3.1 shows the begin and end times of each word but does not indicate the ranges associated with these times. Because of inaccuracies associated with word boundaries, SASS must tolerate some temporal overlap between adjacent words, such as ME and ABOUT in Figure 3.1. Similarly, SASS must skip over gaps or silences, such as the separation between the end of ABOUT and the beginning of BEEF in our example. A truly robust speech understanding system should be able to skip over recognized interjections as in "Tell me about, let's see, beef" and unintelligible intervals as in "Tell me about (mumble) beef." The latter phenomena are not currently tolerated by HEARSAY-II but mechanisms for handling them within the current framework of HEARSAY-II are being investigated.

Finally, SASS must help control the growth of the search space. One way to do this is so operate upon only those word hypotheses whose validity exceeds some minimum "recognizability" threshold. This can be represented visually by covering up all the hypotheses in Figure 3.1 with validity below a threshold of, say, 60. This mechanism reduces the size of the search space at the risk of ignoring correct but poorly rated word hypotheses. Thus SASS should be able to increase its sensitivity dynamically to detect hypotheses previously ignored. SASS should also be able to fill in missing words, i.e., expand the search space. Figure 3.2 shows the words predicted by SASS on the basis of the words recognized in Figure 3.1. Note that many incorrect predictions are made in addition to the correct word TELL. This is because (1) SASS has no way of knowing which recognized words are correct and, hence, makes many predictions on the basis of incorrect cues; and (2) many incorrect predictions may be made from a recognized cue, i.e., many words may be syntactically and semantically plausible in a given context. The number of words plausible in a given context is called the *branching factor*. The reliability of a prediction is inversely related to its branching factor. In order to avoid exploding the size of the search space, SASS must take into account the variable reliability of different inferences and generate only the most reliable predictions.

In sum, SASS must help guide the search for a solution by recognizing plausible word sequences ("islands of reliability") and using them as a basis for filling in missing words. In this process, the search space must be explored in a best-first order and be expanded cautiously on the basis of reliable evidence.

4 *OVERVIEW OF THE CHAPTER: THE EVOLUTION OF SASS*

The evolution of SASS can be regarded as an exploration of various recognition behaviors derivable from linguistic knowledge. The language recognized by HEARSAY-II can be described by a set of *templates* -- sequences of word and phrase categories -- occurring in that language. This set of templates is called a *semantic template grammar* [1,3,4]. As an example, the template named $REQUEST is defined as the sequence TELL $ME $RE $TOPICS and describes a set of typical requests which might be made to a news retrieval system. An example of a $REQUEST is the utterance TELL ME ABOUT BEEF. The dollar sign prefix indicates a category; e.g., $ME is the category {ME, US}. Categories may contain words, phrases, and templates; for example, the template $TOPICS is defined recursively to include sequences of the form $TOPICS OR $TOPICS and $TOPICS AND $TOPICS. The linguistic knowledge used in SASS is described in detail in Section 5.

The definition of $REQUEST represents the knowledge that "the template $REQUEST occurs in the language and consists of the word TELL, followed by (a member of the category) $ME, followed by $RE, followed by $TOPICS." How can this knowledge be applied to produce behavior rules that will help HEARSAY-II detect utterances described by the template? One obvious type of behavior is recognition. The recognition rule for $REQUEST can be stated as "Given temporally adjacent hypotheses for TELL, $ME, $RE, and $TOPICS, hypothesize that $REQUEST occurs in the time interval starting at the beginning of the hypothesized TELL and ending at the end of the hypothesis for $TOPICS." Note that TELL is an hypothesis at the word level of the blackboard, while $ME, $RE, $TOPICS, and $REQUEST are hypotheses at the phrasal level. A

mechanism for performing such recognition efficiently is described in section 6.

While this mechanism succeeds in efficiently identifying grammatical word sequences in a combinatorial search space of possible sequences, it is inadequate to find the correct utterance unless the search space is complete, i.e., contains all the words in the utterance. Since the search space provided by bottom-up word recognition is generally incomplete, it is necessary to implement some mechanism for predicting missing words.

Predictive behavior can be inferred from the linguistic knowledge contained in the grammar. For example, from our definition of $REQUEST, we can generate prediction rules such as "Given an hypothesis for TELL, hypothesize that $ME occurs to its right (i.e., following it in time)" and "Given an hypothesis for $ME, hypothesize that TELL occurs to its left." Obviously such rules should only be applied to predictive cues having bottom-up (acoustic) support, lest unverified predictions are used as bases for further predictions. Thus predictions must be evaluated by lower-level KSs that can match them against the acoustic data and rate them. Note that the first example of a prediction rules generates an hypothesis for $ME, which is not a word. In order for an hypothesized $ME to be evaluated, it must be *respelled* into hypotheses for its constituent words, which can then be rated by lower-level KSs. Thus the knowledge that $ME = {ME, US} leads to *respelling rules*: "Given an hypothesis for $ME, hypothesize that ME occurs in the same time interval" and "Given an hypothesis for $ME, hypothesize that US occurs in the same time interval." The hypotheses for ME and US represent alternative interpretations of the same time interval of the utterance. We can see that the hypothesization of a category containing many elements corresponds to a large branching factor prediction. Prediction and respelling are described in section 7.

Prediction is a weak inference method, because it generally leads to many incorrect hypotheses and a concomitant explosion of the search space. Thus only the most reliable predictions should be made. For example, the prediction of TELL is much more likely to be correct if $ME $RE $TOPICS has been recognized than if only $ME has been recognized. Thus it would be useful to have a prediction rule, "Given temporally adjacent hypotheses for $ME, $RE, and $TOPICS, hypothesize that TELL occurs to the left of $ME." In order to use such a rule, a mechanism is needed for recognizing when the condition of the [condition→action] rule is satisfied. However, the sequence $ME $RE $TOPICS is not a template in the grammar, so it is not detected by our recognition mechanism. A template for $ME $RE $TOPICS could be added to the grammar, but this is only a single subsequence of $REQUEST. $REQUEST has an infinite number of possible word subsequences, so it is hardly feasible to generate templates for every one. Thus we need a more general device for recognizing partial instantiations of templates. For example, the definition of $REQUEST can be used to generate the following *partial recognition* rule: "Given temporally adjacent hypotheses for any contiguous subsequence of TELL $ME $RE $TOPICS, hypothesize that $REQUEST occurs starting at or before the beginning of the first hypothesis and ending at or after the end of the last hypothesis." Prediction of the template's missing constituents is then effected by a *reconstruction rule* which is a modified form of respelling: "Given an hypothesis for $REQUEST which is supported by some but not all of its constituents TELL, $ME, $RE, and $TOPICS, hypothesize the missing constituents in the appropriate time intervals." Partial recognition and reconstruction are sufficient to replace prediction. Partial-matching, described in section 8, is theoretically sufficient to identify any grammatical subsequence of recognized words. Once a general partial-matching mechanism is available, SASS could restrict itself to predictions made from only the most reliable (longest and best-rated) subsequences, so as to avoid exploding the search space with large numbers of predictions based on unreliable evidence.

Unfortunately, our implementation of partial matching produced a combinatorial explosion in activity, since the conditions for recognition of a template were relaxed to tolerate the absence of necessary constitutents. Thus partial recognition rules were not effective in filtering out noisy data, and a new filtering mechanism had to be devised. The mechanism adopted was a word sequence detector named WOSEQ that efficiently identifies word sequences which are potentially realiable (high-rated and relatively complete) partial matches of templates. WOSEQ builds word sequences out of word hypotheses that are pairwise temporally and grammatically adjacent. An ordered pair of words is considered grammatically adjacent if the second word can follow the first word in some utterance in the language. Grammatical adjacency information is precomputed from the grammar and stored in a boolean matrix. Thus the grammatical adjacency of a given pair of word hypotheses can be determined by a simple retrieval, without an expensive search through the grammar. The word sequences hypothesized by WOSEQ are only pairwise grammatical and must therefore be parsed by SASS. A word sequence which is parsed, i.e., recognized as a partial instantiation of a template, can then be used as a reliable basis for prediction of missing constitutents. This filtered partial matching scheme is described in section 9.

We have seen that linguistic knowledge can be converted into several types of hypothesize-and-test rules. The strong rules tend to convert knowledge into the form of tests, while weaker rules use the same knowledge for generating new hypotheses. For example, the recognition rule for a sequence template such as $REQUEST tests that hypothesis for all sequence constituents present and temporally adjacent. In contrast, a prediction rule for the same template tests the presence of only a single constituent (e.g., TELL), and uses knowledge of the sequential nature of the template to generate an hypothesis for an adjacent constituent ($ME).

Although the current version of SASS has been quite successful, it has certain limitations. These limitations and other issues our work has highlighted are discussed in section 10. The effectiveness of the current scheme is evaluated in section 11, and the conclusions of our experience with SASS are summarized in section 12.

5 SYNTACTIC AND SEMANTIC KNOWLEDGE

Syntactic and semantic knowledge in SASS is supplied by a task-specific *semantic template grammar*, which can be characterized as a context-free grammar with ambiguity and recursion permitted. The templates (nonterminals) of the grammar are sequences and categories representing sets of phrases in the task language. Although the grammar is a context-free grammar in form, as are general English phrase structure grammars, there is an important difference in the spirit in which this form is used. A semantic template represents a set of semantically equivalent (or similar) phrases. In contrast, a nonterminal in a general phrase structure grammar represents a set of phrases which can have the same syntactic role (e.g., <noun phrase>) but are otherwise unrelated. Consequently, a phrase structure grammar can generate -- and parse -- utterances that are syntactically permissible but semantically nonsensical. Conditions for semantic consistency must be represented as additional features imposed on the grammar, such as the augments in augmented transition networks [19]. This is not necessary in an appropriately-designed semantic template grammar. Thus a semantic template grammar imposes considerably more constraint on the phrases it accepts. This constraint is crucial in rejecting many of the incorrect sequences of words hypothesized by an imperfect word recognizer in a speech understanding system. Moreover, the use of a semantic template grammar means that much of the semantic analysis of an utterance is accomplished by the process of parsing it. In particular, the parse of an utterance can classify it as to its general intention.

The concept of semantic template grammars is largely based on the PARRY system, which simulates a paranoid patient being interviewed [1]. PARRY's linguistic knowledge consists of 1700 templates representing standard types of questions abstracted from transcripts of psychiatric interviews. For example, one such template is (WHAT BE YOU JOB). Each component of this sequential template is the canonical member of a class of semantically equivalent words. An utterance such as "What is your occupation" is processed by replacing each word by its canonical equivalent, yielding "WHAT BE YOU JOB." This matches the template, whose associated semantic interpretation is then used to determine an appropriate response. Colby's template-based parsing scheme is considerably more sophisticated than this example shows. Its significance with respect to the current discussion is its ability to interpret a typed utterance in unconstrained English in under one second. Although such a scheme must be modified to cope with errorful input (uncertain word hypotheses) and ambiguity (of words and phrases in more than one semantic class), a semantic template grammar can provide considerable efficiency in the analysis of a spoken utterance.

In a system with perfect input, the grammar is used in a straightforward way to parse the input word sequence. In contrast, a speech understanding system must use its grammar in more sophisticated ways, since the correct word sequence must be identified amid many incorrect word hypotheses and on the absence of some correct words.

One use of syntactic and semantic knowledge, hereafter referred to simply as *grammatical knowledge*, is in finding grammatical sequences of temporally contiguous hypothesized words. This process is called *recognition* and corresponds to parsing, since a potential sequence must be parsed to determine whether it's grammatical.

Another application of grammatical knowledge is to hypothesize constituents which are likely in the recognized context but have not been hypothesized bottom-up. This process, called *prediction*, is based on the notion that if part of a sequence template is found, it's a good idea to look for the rest of it.

A third application of grammatical knowledge is *goal reduction*. Prediction can be considered as generating the goal of finding the predicted phrasal constituent. This goal must be broken down into subgoals for finding individual words before the lower level KSs can be applied. This form of goal reduction consists of respelling a predicted sequence (conjunctive goal) into hypotheses for its individual sequence elements and respelling a predicted category (disjunctive goal) by enumerating the members of the category and hypothesizing each one.

Grammatical knowledge can be used in still another way, to reduce uncertainty among hypotheses by identifying hypotheses for which there is confirming contextual evidence. This process is called *postdiction* and is based on the notion that the plausibility of an hypothesized phrase is higher if it is adjacent to a phrase grammatically consistent with it.

Recognition, prediction, respelling, and postdiction are just some of the applications of grammatical knowledge. All four are derivable from the same templates; some of them (recognition and postdiction) test consistency (temporal and grammatical adjacency) of a template's hypothesized constituents, while others (prediction and respelling) use the hypothesized constituents as evidence for hypothesizing missing constituents. In terms of the generate-and-test paradigm, strong methods are characterized by using knowledge as a basis for testing, while weak methods use the same knowledge for generation.

These generic types of problem-solving behavior based on grammatical knowledge can be represented as [stimulus→response] productions [14]. Given a template and its constituents, it is straightforward to derive productions for recognizing the template when all of its constituents are recognized, predicting and postdicting each constituent from adjacent constituents, and respelling a predicted template into its constituents. This derivation is performed by a program called CVSNET (Convert Semantic Net), which compiles the grammar into the various types of productions. Thus the run-time component of SASS need only be a production system interpreter to identify activated productions and perform the appropriate generic behavior associated with

each one. Furthermore, new generic productions can be introduced simply by defining their procedural behavior in SASS and augmenting CVSNET to derive them. Since the behavior is inferred from the knowledge implicit in the grammar, no change in the grammar itself is required. This is not the case in systems that represent behavior more explicitly in the grammar. Finally, the non-procedural representation of grammatical knowledge greatly simplifies its augmentation, since semantic template grammars are easy to understand, create, and modify.

6 BOTTOM-UP PARSING AND ACORNs

6.1 Problem

Given the semantic template grammar, the purpose of bottom-up parsing is to find a phrase (non-terminal), if it exists, that generates the sequence of words under consideration. Programming language compilers such as Algol W and Pascal expect the input to be syntactically correct and use various parsing techniques, such as LR(k), precedence grammars and recursive descent, to parse the input from left to right. The existence of an error usually results in an informationless error message and the abortion of the compilation. Recent work in error-correcting parsing [17] has barely scratched the surface with respect to the error recovery capabilities needed in a speech understanding system.

The SASS knowledge source is faced with numerous problems when parsing in recognition mode. The information that is extracted from the utterance by other KSs is highly errorful. This error is characterized by substitution, deletion and insertion of words throughout the utterance. At present, about ten incorrect words are hypothesized with each correct word. Thus SASS is faced with the problem of weeding out the incorrect information so that the correct words can be identified. Frequently an incorrect word sequence is parsed during the search for the correct word sequence. Though the sequence being parsed may be incorrect, it may contain a subsequence of the correct sentence. Since incorrect word sequences are frequently processed for awhile and then discarded and many of these sequences contain common subsequences, it would be good to avoid the large amount of recomputation entailed in parsing the common word subsequences.

A third problem that must be dealt with is the avoidance of a pure depth-first search for the correct sentence. Strict left-to-right parsing is a depth-first search in the sense that is it committed to the word sequence being extended. The extension of a sequence is terminated only when it has accumulated a low validity. The amount of computation needed to recognize and terminate an incorrect sequence can be quite large, especially when the initial subsequence is correct. Thus, it seems necessary to work with the best information available at all times, no matter where it occurs in the utterance.

6.2 Solution

In order to solve these problems, the ACORN model was developed [3]. An ACORN

(Automatically Compilable Recognition Network) can be described simply as a filter network where information enters via terminal nodes (leaves) and flows through the network, each node filtering the information that traverses it.

An ACORN is constructed for the semantic template grammar described in the previous section. Terminal nodes in the ACORN correspond to words in the grammar; nonterminal nodes correspond to phrases (templates) in the grammar. A directed arc connects a template (source node) to the template (sink node) it supports. The direction of the arc defines the flow of information from the source template to the sink template. Each node defines a filter upon the information that reaches it. The template grammar ACORN has two types of filters: conjunctive and disjunctive. The conjunctive filter requires information from all its supporting nodes to be present and satisfy a specified relation before information is passed to a higher node. It collects information until all is present, then passes it on. The disjunctive filter requires information from at least one supporting node in order to pass information on. A node is instantiated whenever its relation is satisfied.

When a word is hypothesized on the blackboard, the terminal node that corresponds to the word is instantiated. The word's positional information and validity enter the ACORN via the terminal node and are passed along the arc joining the terminal word node to the template node it supports. This positional information is stored at the supported template and if the template node is disjunctive the information is again passed on. But if the template node is conjunctive then the positional information remains at the node until information from all nodes that support this template node has arrived and satisfies the specified relation. Once the relation is satisfied, a composition of the information is passed on. The information flows up through the network towards the root. Most paths of information terminate far from the root due to the unsatisfiability of a template node relation. Thus large amounts of information (word phrases) are filtered due to the lack of supporting information (words or phrases that can be joined with the phrase). The paths of information that do combine to reach the root node define a set of candidate sentences and their corresponding derivation trees.

All information that enters and is stored in the ACORN remains throughout the life of the ACORN (i.e., the analysis of a particular utterance). This is an important characteristic. Once a subphrase has been parsed, it is never parsed again regardless of the number of different phrases subsuming it. This is a consequence of the existence of paths in the ACORN from the node defining the subphrase to all subsuming phrases. Information about the instantiation of the subphrase template is known to all higher phrase templates that use it (i.e., a template that has the subphrase as a direct son). Thus the results of the recognition of various words and their constructed phrases are permanently stored and continually used in the ACORN.

In reality, the constructed ACORN is not isomorphic to the template grammar.

There exists an intermediate phase in the construction process in which the grammar is canonically decomposed into binary productions. As a result, there is only one instance of each different template in the ACORN and it is directly connected to each template of which it is a subphrase. In addition each conjunctive node in the network is supported by only 2 nodes, simplifying the storage and testing of information and minimizing the amount of recomputation.

The construction and use of the ACORN in HEARSAY-II will be elucidated by an example. Figure 6.1 defines an amended (for expository purposes) portion of the template grammar used for a news retrieval task. Only the nonterminals used for the derivation of the sentence

TELL ME ABOUT BEEF

are shown. The sentence symbol for the grammar is the nonterminal $PARSE. All sentences in the grammar are derived from it. Disjunctive templates employ set brackets to define templates that directly support them (e.g., $REQUEST). Productions without set brackets define conjunctive templates.

The information that enters the ACORN from the word hypothesis includes its begin and end times and validity. A disjunctive node in the ACORN passes on the positional information and validity that reach it from its supporting templates. A conjunctive node, once all the the supporting templates are instantiated, tests its relation. If a test is true, the corresponding information is passed on. A template can be instantiated by more than one piece of information, i.e., the occurrence of the same word twice in an utterance will twice instantiate the same terminal word node in the ACORN with two different sets of information. The test that is applied at a node must be applied to all possible combinations of information from the supporting templates. The relation is an adjacency test of the words and phrases that support the template. The ordering of the word and phrase templates is defined by the ordering of nonterminals on the right side of a production in the template grammar. In other words,

$LOOKUP+TOPIC → $LOOKUP $TOPIC

defines the $LOOKUP+TOPIC template node (nonterminal) to be supported by the templates (nonterminals) $LOOKUP and $TOPIC, in that order. The adjacency test for the $LOOKUP+TOPIC template is invoked when both the $LOOKUP and $TOPIC templates are instantiated. The positional information of both is tested to see if $LOOKUP is left-adjacent (prior) to $TOPIC. If the adjacency test succeeds, the begin time from the leftmost (earliest) supporting template and the end time of the rightmost (latest) supporting template are used as the positional information of the $LOOKUP+TOPIC template. The new times and a function of the supporting validities are passed up the network. If the test fails, no information is passed along. It is important to note that the successful instantiation of a template implies the successful parsing of the supporting word sequence.

```
1      $PARSE → [  $REQUEST+]
2      $REQUEST+]            →      $REQUEST  ]
3      $REQUEST                     →
               {             $LOOKUP+TOPIC
                             $BE+TOPIC
                             $BE+TOPIC+IN+THE+NEWS  }
4      $LOOKUP+TOPIC         →     $LOOKUP $TOPIC
5      $LOOKUP               →
               {             $GIMME+RE
                             $GIMME+SOMUCH+RE  }
6      $GIMME+RE             →     $GIMME  $RE
7      $OUTPUT               →
               {             GIVE
                             TELL     }
8      $TO+USER                     →
               {             ME
                             US       }
9      $OUTPUT+TO+USER       →     $OUTPUT  $TO+USER
10     $GIMME                →
               {             $IWANNA+RECEIVE
                             $OUTPUT+TO+USER
                             $AUX+YOU+OUTPUT+TO+USER
                             $WHAT+BE
                             $BE+THERE         }
11     $RE!WORD                     →
               {             ABOUT
                             ON       }
12     $RE                   →
               {             $RE!WORD
                             $RE!WORD+REL!PRO        }
13     $TOPIC                →
               {             $WHO
                             $WHAT!
                             $WHERE
                             $THE+WHAT!        }
14     $WHAT!                →
               {             BEEF
                             ROBOTS
                             GOLF
                             FOOTBALL        }
```

Figure 6.1: Partial News Retrieval Template Grammar

The adjacency test must be able to cope with several problems inherent in speech:

1) boundary fuzziness -- imperfect adjacencies;
2) silence -- segments of silence between words;
3) useless interjections -- semantically meaningless interjections;
4) gaps -- short periods of uninterpretable speech.

These errors occur often in actual data. Methods that are robust (i.e., insensitive to data errors) and efficient must be used to handle these errors in a natural way. To handle boundary fuzziness, an inexact adjacenty test is used that allows a parameterized amount of overlap or separation. Useless interjections and silences are accepted by treating each interjection as a silence and allowing the adjacency test to succeed if the two words or phrases are separated by a silence. If two phrases are separated by a short period of uninterpretable speech and if the separation is greater than the allowable fuzziness separation and less than the maximum gap separation, a *gap hypothesis* is generated for the separation and the adjacency test succeeds. The validity of the hypothesis is a decreasing function of the gap duration. This method does not solve the problem,currently under investigation, of deciding when an interval of speech is uninterpretable.

Figure 6.2 shows the ACORN corresponding to 6.1, with the sentence "TELL ME ABOUT BEEF" being recognized.

Information about the word hypotheses TELL, ME, ABOUT and BEEF enters the ACORN via the corresponding terminal templates. The information is of the form (begin time: end time, validity). The $OUTPUT template is instantiated with (0:10,80) from the word hypothesis TELL. This information tuple is immediately passed to (instantiates) the $OUTPUT+TO+USER template because $OUTPUT is a disjunctive template. Here the tuple remains until the rest of the constituents of this template are instantiated. Since ME is a word hypothesis, the $TO+USER template is instantiated with the tuple (15:20,60). The adjacency test is applied to the information supporting the $OUTPUT+TO+USER template resulting in its instantiation with (0:20,70). The $GIMME template is now fully instantiated. The adjacency test is applied to the $GIMME and $RE templates. Assuming that a separation of five units is allowable under the fuzziness boundary, the adjacency test succeeds and a new tuple, (0:40,73), is composed of the begin time of the $GIMME template instantiation and the end time of the $RE template instantiation and an average of both their validities. This new tuple is used to instantiate the $GIMME+RE template and is also passed on to the $LOOKUP template. The process continues similarly until all word hypothesis information has entered and flowed through the ACORN.

In the HEARSAY-II environment, each instantiation of a template is accomplished by a separate execution of the SASS knowledge source. The scheduling execution is controlled by the focussing strategy of the system. If the information exists to instantiate more than one template, the template supported by the highest rated information will be instantiated first.

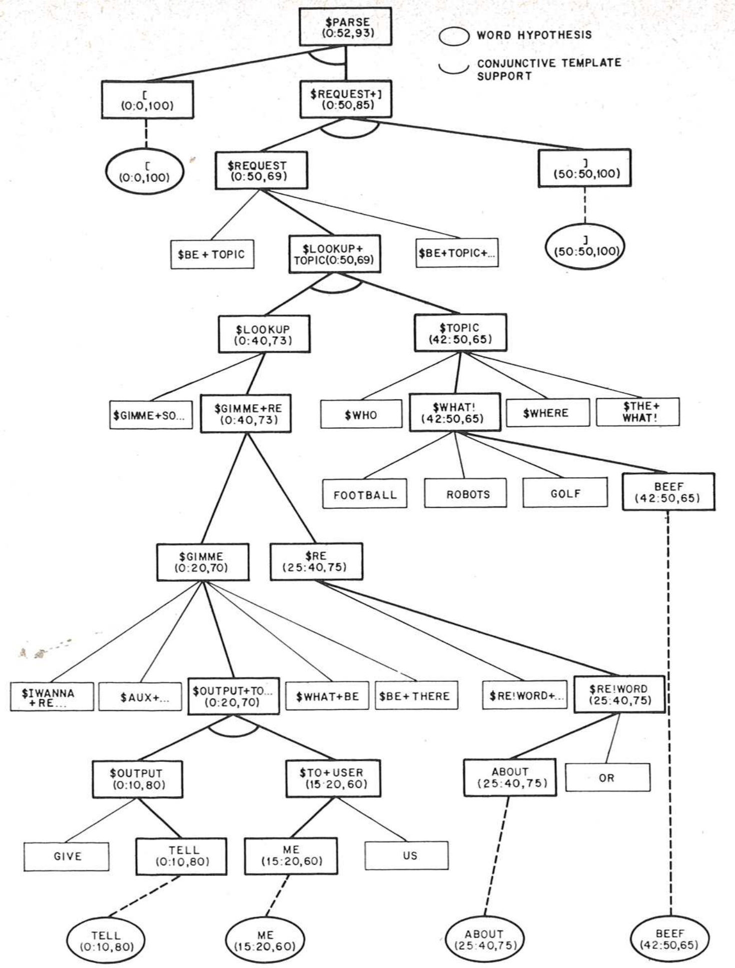

Figure 6.2: ACORN recognition of "[TELL ME ABOUT BEEF]"

The template instantiation process consists of creating a phrasal hypothesis on the blackboard for the template. The begin and end time and validity information are derived from the hypotheses that support the new phrasal hypothesis. Links are also placed on the board, joining the new hypothesis with its supporting hypotheses. Figure 6.3 shows the blackboard structure that would be built for recognizing TELL ME ABOUT BEEF.

Only the instantiated portions of the ACORN are placed on the board. Note that in the description of information entering the ACORN, the terminal word nodes were instantiated by word level hypotheses on the blackboard. In the nodes of the ACORN instantiated at the phrasal level on the blackboard, however, the terminal nodes of the ACORN do not appear since each word level hypothesis can be directly linked to the instantiated father template of its corresponding ACORN terminal node.

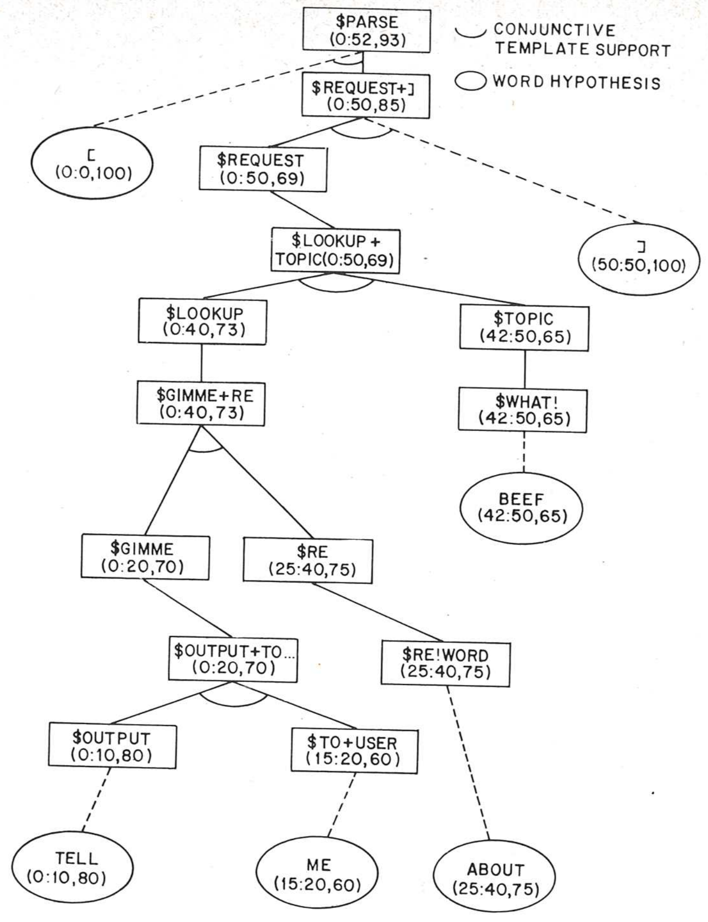

Figure 6.3: Blackboard representation of ACORN recognition of "[TELL ME ABOUT BEEF]"

A sentence is successfully parsed and recognized when the sentence symbol of the semantic template grammar is instantiated. The words constituting the recognized sentence can be found by tracing through the instantiated templates that support the sentence symbol template. The terminal instantiated templates in the ACORN are the words of the sentence.

The use of the ACORN in recognition mode is limited in the sense that it requires all information (i.e., all words) necessary to parse the correct sentence to be hypothesized by the lower modules. But present statistics show that only 80% of the words in the utterance are supplied to the ACORN. Therefore other methods must be utilized to provide the missing information.

7 PREDICTION

7.1 Problem

Although the ACORN can effectively filter out noise contained in data provided by lower-level modules, it is unable to recognize the correct utterance unless all the correct words and their correct times are provided. In HEARSAY-II, there is no guarantee that all correct words will be hypothesized. Speech problems such as the omission of a syllable at word junctures, background noise or poor speaker enunciation result in incorrect words being hypothesized with high ratings. The correct word may be hypothesized but with a low validity or it may not be hypothesized at all. Even if the correct word is hypothesized, it may never be worked on by SASS knowledge; the focussing policy of the system directs attention to the highest rated words. Thus a correct word with low validity may never be considered unless the alternatives are exhausted.

7.2 *Solution*

The solution to this problem of missing or poorly rated constituents is to use phrases recognized by the ACORN as "islands of reliability" from which to predict constituents that could precede or follow these phrases. In other words, the instantiation of a semantic template in the ACORN is considered to provide enough information for the system to predict what words can possibly precede or follow the word sequence supporting the template. The longer the island of reliability, the better, in terms of validity, is the prediction. This is a result of the island having a grammatical constraint directly related to its length.

The prediction process requires the existence of a seed, a maximally instantiated semantic template in the ACORN. A maximally instantiated template is the root of a subtree representing the parse of a recognized word sequence. In other words it is the highest template in the ACORN instantiated by the recognition of a word sequence. Actually, a word sequence can give rise to more than one subtree depending on the structure of the ACORN. Figure 7.1 shows the part of the ACORN instantiated on the blackboard by the recognition of the words "TELL ME". $GIMME is the template instantiated by the word sequence TELL ME. It is used as the seed. The prediction process instantiates templates that are grammatically adjacent to the seed template (Figure 7.2). This implies that the leftmost or rightmost word support of this *predictively instantiated template* (PIT) is adjacent to the seed's supporting word sequence.

Respelling is the next step in the prediction process. To respell is to take a missing constituent of a PIT and instantiate the part of the ACORN necessary to hypothesize words adjacent to the seed. In other words, the ACORN is instantiated downward on a side adjacent to the seed. Figure 7.3 shows the predictive instantiation necessary to hypothesize the words ABOUT and ON which can be right adjacent to the word sequence. The templates $RE, $RE!WORD, ABOUT and ON are also PITs. The terminal word templates ABOUT and ON are now available for verification by other modules in the HEARSAY-II system.

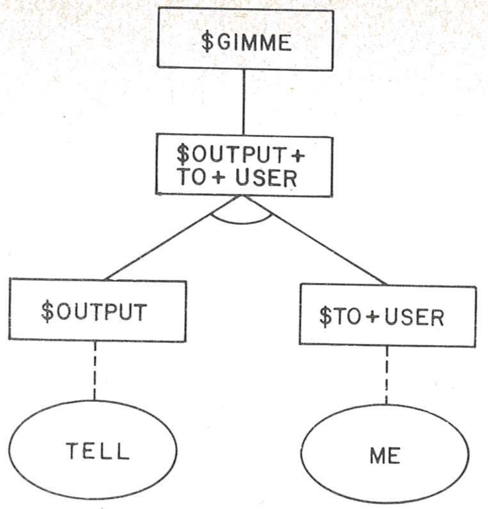

Figure 7.1

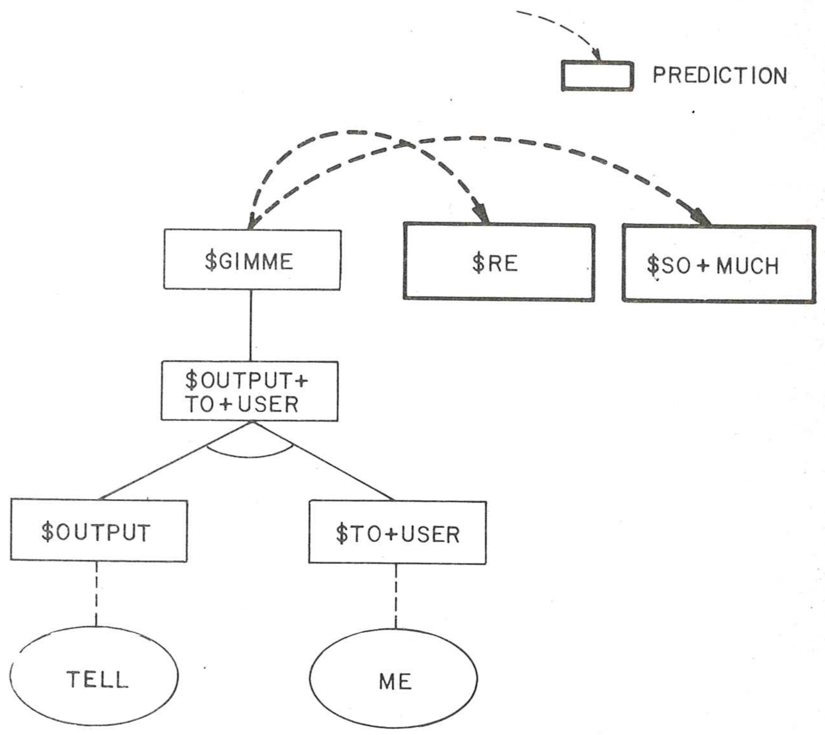

Figure 7.2

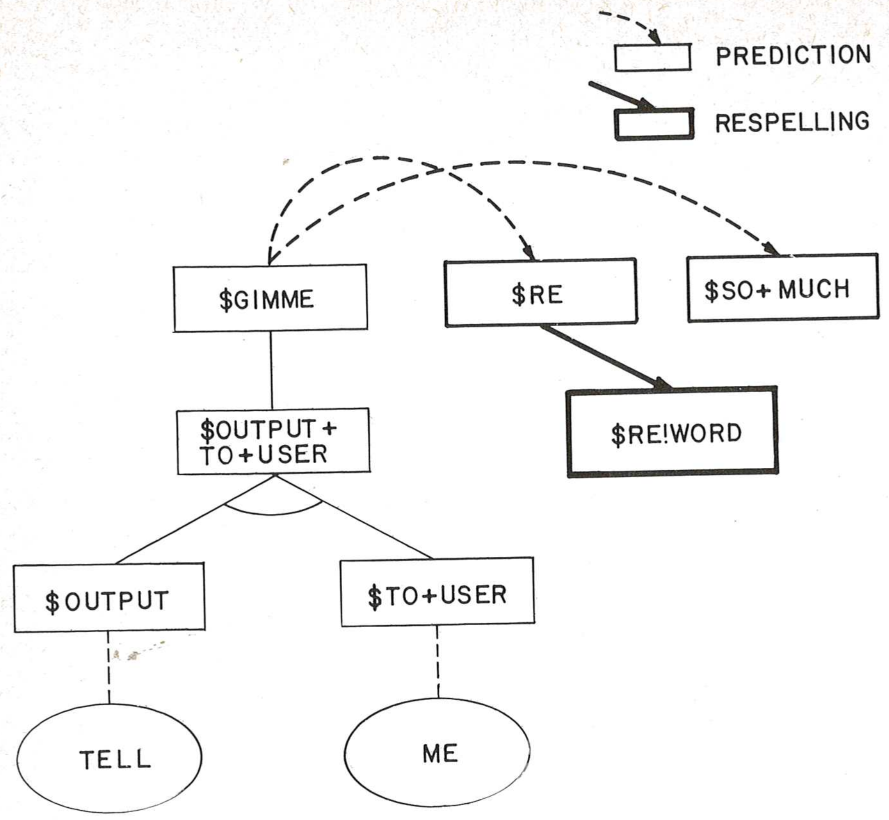

Figure 7.3

Actually, the respelling process used differs slightly from the example. Since $RE!WORD is a disjunctive template whose sons are all word templates, the terminal templates are not hypothesized. Instead the verifying KSs utilize knowledge of which words are included in the $RE!WORD set. This is important for efficiency. In a large vocabulary, the number of words in that set can be quite large and the blackboard hypothesization overhead expensive. If a word in the $RE!WORD prediction set is verified, the verifying KS can hypothesize the word. This new word would be recognized by SASS, the corresponding word template would be instantiated, the word hypothesis and template would be linked, and the terminal template would be inserted into the instantiated ACORN (blackboard structure) by linking it to the PIT $RE!WORD. An insertion is characterized by the linking of a new instantiated (hypothesized) template to an existing instantiated template on the blackboard.

7.3 *Difficulties and Refinements*

Combinatorial problems arise when prediction is insufficiently restricted. For a number of reasons, many of the predictions that occur will be incorrect. With a vocabulary of 1000 words, the number of words hypothesized bottom-up is quite large. As a result the number of recognized phrases is large also. Even if only two words are predicted from each of these phrases, one on each side of the phrase, the number of predicted words is quite large. Compounding this problem is the branching factor of the grammar. The average number of words which can precede or follow a given phrase is considerably greater than one, even for a highly constrained language. The various task grammars used by HEARSAY-II have branching factors of five, ten, or more.

The prediction explosion can be controlled through the use of thresholds. Thresholding is applied in two separate ways. The validities of predictive phrases are thresholded. If validity of a phrase is below a specified threshold then no predictions will be made from it. Another type of thresholding is based on branching factors. If the number of words that can be predicted from a phrase is above a specified threshold, prediction is not allowed. In both cases the thresholds can be dynamically modified, if desired, to reflect variations in the state of the recognition process in different temporal regions of the utterance. Although thresholding constrains the combinatorics of prediction, the amount of constraint afforded is not enough. When excessive thresholding is applied, the correct words are not predicted. The degree of laxity in the thresholds necessary to avoid this problem causes a combinatorial explosion in the number of predictions that occur.

8 *PARTIAL MATCHING*

8.1 *Problem and Solutions*

It is necessary in the HEARSAY-II environment to execute an action only when it is well supported by blackboard evidence. If this rule is not followed at all levels, the system quickly degenerates. This maxim is especially applicable to prediction. There exists a large number of possible phrases from which to predict; the system must be confident of a phrase before it is to be used. This implies that only the most reliably supported templates should be used for prediction. However, a phrase -- a grammatical sequence of recognized words -- may not directly instantiate any template. Instead it may partially instantiate one or more adjacent templates leaving several constituents unsupported. If only fully instantiated templates were used as predictors, predictions would be made without the benefit of using the most reliable phrases. The predictors used would be subsequences of these phrases corresponding to fully instantiated templates, and they would be less reliable. To overcome these defects, HEARSAY-II recognizes and instantiates partially matched templates and uses them as a basis for prediction.

Taking the example of the recognition of the words ME ABOUT, Figure 8.1 shows the instantiated portion of the ACORN produced on the blackboard. The template $OUTPUT+TO+USER is partial-matched. The grounded arcs in the figure designate that a constituent ($OUTPUT) is missing. By partial-matching the template $OUTPUT+TO+USER, prediction, as described in the previous section, has been eliminated. All that has to be done to achieve the same effect as prediction is to fill in and respell the missing constituents of the template. Figure 8.2 depicts this respelling. Only the respelling necessary to derive the word TELL is shown.

8.2 *Difficulties and Refinements*

The introduction of partial matching into the HEARSAY-II system immediately met with difficulties. The possible number of partial matches was enormous. Combinatorial explosion occurred. Suddenly the sky darkened, and it began raining hypotheses and links. To counter the combinatorics, thresholding was introduced. Phrases with partial support were not recognized (matched) unless the validity of the support was high.

8.3 *Evaluation and Limitations*

The introduction of such thresholding served to reduce the number of partial matches identified. This constraint was inadequate though to eliminate the combinatorial explosion entailed by relaxing the earlier requirement that all constituents of a template be present in order for it to be instantitated on the blackboard.

From testing these ideas we have drawn the following conclusion. Partial matching results in a combinatorial explosion that simple thresholding cannot control. Since so many partial instantiations were possible -- most of them based on incorrect data -- it became clear that some mechanism was needed that could find only the best (i.e., most complete and reliable) partial matches.

9 *FILTERING AND PARTIAL-MATCHING ACTIVITY*

9.1 *Problem*

In the previous section, partial matching was introduced as a method for recognizing maximal grammatical subsequences to be used for prediction. Because the partial-match mechanism was applied to all lexical data, the amount of partial matching grew out of control. The introduction of thresholding did not remedy this problem. The root of the problem was twofold. First, single word hypotheses are too unreliabl to warrant partial matches. Second even using correct hypotheses, too many partial matches were unnecessarily made along with the desired maximal partial matches. It was necessary to constrain the partial-match process to operate only on reliable (lon highly rated) word sequences.

9.2 *Solution*

An analysis of the problem resulted in the development of a KS called WOSEQ to identify the most promising word sequences by using a relatively inexpensive test.

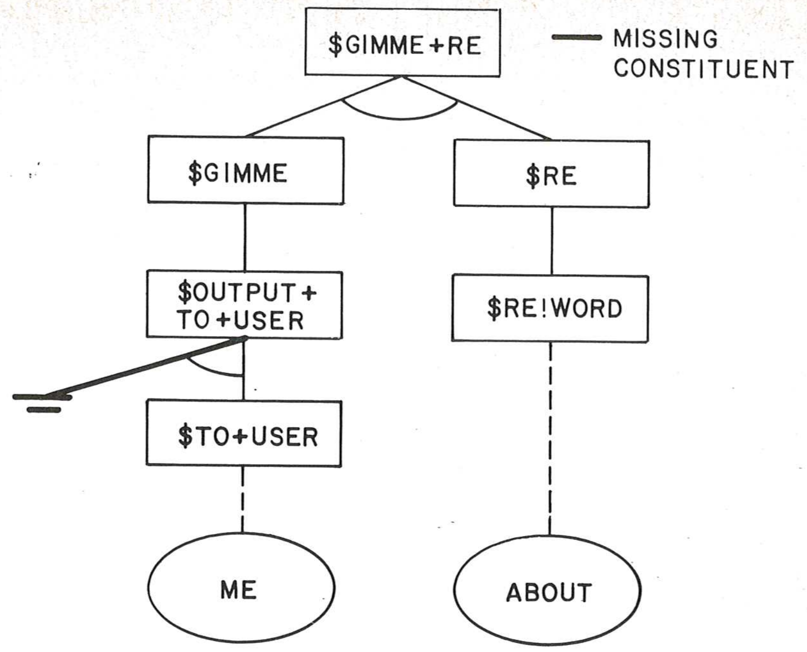

Figure 8.1: Partial match of sequence "ME ABOUT"

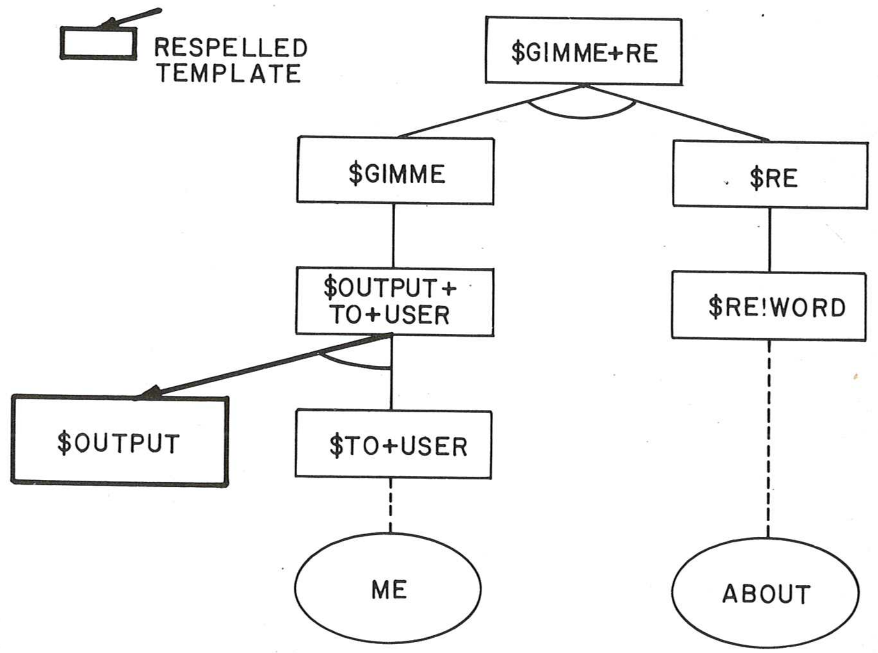

Figure 8.2: Respelling of missing constituents of $OUTPUT+TO+USER

9.4 Evaluation

The introduction of the WOSEQ module into the HEARSAY-II system succeeded in constraining the combinatorial explosion that had previously plagued SASS. Although WOSEQ generates only a few sequences, one or more of them are usually correct or contain correct subsequences. In the latter case, the decomposition mechanism eventually finds the correct subsequence for SASS. By permitting partial matching to be applied only to a small set of reliable sequences, the search space has been drastically reduced.

9.5 Limitations

WOSEQ knowledge is based on pairwise grammatical adjacency. The module is limited in that all constitutents of a sequence must be contiguous. Thus WOSEQ cannot identify good, potential partial matches containing holes, e.g., TELL ME ... BEEF.

10 OTHER ISSUES

SASS, as a knowledge source in the HEARSAY-II system, contributes its own specialized knowledge to the fulfillment of the system's overall goal of recognizing a spoken utterance. In order to effect this contribution efficiently and effectively, an analysis of the amount of needed information must be made. For example, many kind of semantic information can be accumulated, stored, and possibly manipulated during the speech recognition process. But there exists a point of diminishing return beyon which further semantic processing will increase the cost of evaluating an hypothesis without contributing to its validation. Hence, when adding knowledge to a system an analysis should be made of how much constraint the knowledge will ostensibly provide and what its cost of application will be.

Semantic knowledge in SASS is represented by the semantic template grammar described earlier. The grammar can be made arbitrarily complex to accomodate various types of anticipated dialogues but must remain context-free in form. Our commitment to representing semantic information in a semantic template grammar stems from the belief that the level of semantic processing necessary to achieve task-oriented speec recognition can be carried out most effectively in this framework. Other representations may be more general, but we believe that most types of generality are not necessary for the purposes at hand. All that generality in syntax and semantics appears to achieve is a higher processing cost per hypothesis.

As emphasized above, a knowledge of the cost of a representation, method, actio etc. is important to the efficient solution of realistic problems such as speech understanding. In HEARSAY-II, KSs are the basic structural elements and their instan tiations are the basic actions. The scheduling of KS instantiations is, in itself, an expensive computation. A certain amount of scheduling overhead can be eliminated if an appropriate order of computation is known a priori. The prediction-verification loop illustrates this point. Whenever a set of words is predicted from a word sequence, the appropriate response is to verify those words. Prediction and verification can be an indivisible single action uninterrupted by scheduling priority com-

These promising sequences are hypothesized on the blackboard at a word sequence level inserted between the lexical and phrasal levels.

The inexpensive test is pairwise grammatical adjacency. An ordered pair of words, W1 and W2, are considered grammatically adjacent if the sequence W1 W2 occurs in some sentence generated by the grammar. A word sequence W1,...,Wn is considered pairwise grammatical if each pair, Wi and Wi+1, is grammatically adjacent. Grammatical adjacency information is precomputed from the grammar and stored in a two-dimensional bit matrix for fast retrieval. For the present vocabulary of 1000 words, the word adjacency matrix requires a million bits or about 30K of PDP-10 words, which is not an excessive amount of storage.

Sequences of words are constructed by identifying a few highly rated words as seed sequences and extending these with words that are both temporally and grammatically adjacent. The same process is then applied to the new sequences. The result is a set of varying length sequences. The best sequences are subjected to partial matching (parsing).

The identified sequences are good in the sense that some or all of the words in such a sequence are often correct. This follows from the low probability that randomly hypothesized incorrect words can be connected to form a grammatical sequence. Furthermore, the average validity of a sequence of words is statistically a more reliable indicator of correctness than the validity of a single word hypothesis; i.e., a highly rated WOSEQ hypothesis is more likely to be correct than a highly rated word hypothesis [11]. WOSEQ successfully and efficiently identifies word sequences for partial matching.

9.3 Difficulties and Refinements

A problem arises when WOSEQ constructs sequence hypotheses. Since its knowledge is restricted to pairwise grammatical adjacency, a constructed sequence containing more than two words may not be entirely grammatical. When SASS looks at a sequence hypothesis created by the WOSEQ module, it first parses it to see if it is an allowable sequence. If so the sequence is used to partially match templates whose missing constituents are then respelled as described earlier. If the sequence does not parse, the hypothesis is rejected. However, it is likely to contain a correct subsequence. Accordingly, the WOSEQ module monitors for any rejections of its hypotheses. If a sequence is rejected, it is decomposed. The decomposition of a sequence entails the construction of two or more sequences that are subsequences of the original. These new word sequences can be hypothesized on the blackboard or held in abeyance by WOSEQ depending on their validity and the overall state of the system.

Decomposition is also applied to WOSEQ hypotheses that are successfully parsed by SASS but lead to dead ends when the system attempts to extend them. Thus sequences that are unsuccessful are decomposed in an attempt to utilize any correct information they may contain.

putation. Another example is the parsing of a WOSEQ hypothesis. These word sequences are initially presented to SASS to be tested for grammaticality. Since the goal is to parse the whole sequence, parsing should be a single indivisible action as opposed to a sequence of KS invocations for each template instantiation encountered in the parse.

A third example of an activity that should be viewed as indivisible is the prediction of words by respelling. Each respelling of a template might require one KS instantiation. In order to predict a set of words, a number of KS instantiations is necessary to trace down through the ACORN to the terminal word templates. Since the goal of respelling is to predict words adjacent to a phrase (word sequence), it is appropriate that the prediction of words be carried out by an internal (non-blackboard based) ACORN search which results in a set of word predictions. This search is an indivisible action in the system.

In addition to scheduling costs, the blackboard structures (template instantiations) for respelling are also expensive. The internal ACORN search mentioned above alleviates two problems: scheduling overhead is eliminated by virtue of invoking onl one action; and representation overhead is reduced by refraining from constructing respelling structures on the blackboard. The actual word predictions are placed, as an attribute, on the phrase from which they were predicted. In that form they remain available for verification by other KSs.

11 EVALUATION

It is virtually impossible to evaluate a single HEARSAY-II KS, such as SASS, in isolation. The performance of SASS is very much dependent on the environment provide by the other KSs. Radically different strategies are appropriate in different environments. For example, if bottom-up word recognition generated all the correct words (plus a few incorrect words), a simple left-to-right parse might be the most efficien way to identify the correct utterance. Similarly, in a system with no bottom-up word recognition but very reliable word verification, a left-to-right, top-down parser would be appropriate. In HEARSAY-II, roughly 80% of the correct words are hypothesized bottom-up, with about ten incorrect words generated for every correct word. It is apparent that such an environment dictates a flexible combination of bottom-up and top-down problem-solving methods applied to the best data available, wherever it occu in the utterance. The task of developing such a strategy was complicated by the fact that the performance of the other KSs varied tremendously as they too were developed.

On a smaller scale, the evaluation of the various problem-solving methods used within SASS must be based not on their adequacy or theoretical correctness but on their efficiency and empirically determined utility when used in various combinations For example, postdiction might turn out to be very worthwhile in some knowledge-based inferencing systems, but it was not cost-effective in SASS given the environment in which it was operating. The information contributed by postdiction was mostly redundant, and simply did not justify the expense of computing it. Accordingly, postdiction was eliminated.

Similarly, a scheme to make predictions based on previous discourse was implemented, found not to be cost-effective, and abandoned (at least temporarily). This scheme was implemented in a straightforward fashion: templates whose instantiation was predicted by a discourse module were represented as hypotheses on the blackboard, and hypotheses consistent with these predictions were postdicted by them. Unfortunately, even when the predictions were correct -- i.e., the predicted templates were in fact instantiated -- this scheme failed to justify itself. The predictions were too vague, in two ways. First, the number of ways of instantiating a predicted template was quite large, so a prediction provided rather little information as to which particular words should be sought. Second, these predictions placed no constraint on where in the utterance the predicted template would be instantiated. For example, the discourse module might correctly predict the category template $FOOD, but coult not specify where in the utterance it would be instantiated. Thus incorrect hypotheses representing words in this category were postdicted along with the word BEEF which had been correctly hypothesized at the end of the example utterance, TELL ME ABOUT BEEF. Finally, when the predicted template was instantiated bottom-up anyway, as happened more often than not, the computation associated with discourse prediction was simply redundant, and merely slowed down the recognition process. Other speech understanding systems which use sophisticated semantic inferences appear to suffer from the same problems.

Efficiency considerations resulted in the ACORN instantiation process being made internal to SASS. The only information now placed on the blackboard at the phrasal level is the recognized and extended word sequences, with their maximally partially matched template and missing constitutents as attributes.

The success of the current scheme seems due to the application of three key concepts: validity, sequential constraint, and branching factor. Validity and sequential constraint are used to identify maximally promising phrases. These in turn form the basis for predictions which are scheduled according to their reliability, which is an increasing function of predictor validity and a decreasing function of predictive branching factor. Predicted words that are subsequently found are used to extend sequences and to help determine their revised validity.

An interesting aspect of this scheme is its refinement of the hypothesize-and-test paradigm to make use of partial solutions. Hypotheses representing promising partial solutions (phrases) are expanded into more complete solutions by the processes of prediction, extension, and recognition. Hypotheses representing discredited partial solutions -- ungrammatical sequences or inextensible phrases -- are contracted into less complete solutions by the process of decomposition, in the hope of eliminating incorrect constituents of an otherwise correct partial solution. This mechanism represents a generalized form of backtracking in which the direction of contraction is no longer constrained to be the most recent direction of expansion.

An important property of any large knowledge-based inferencing system is the ease with which knowledge can be modified and extended. Because SASS's problem solving behavior is inferred from its grammar rather than explicitly coded in it, the grammar is simple and readily modifiable. This property became quite noticeable when the HEARSAY-II task was changed from retrieval of Associated Press news stories to retrieval of artificial intelligence abstracts, and the vocabulary was accordingly replaced. Writing a grammar for the new task required only a few man-hours, and did not require modifications to SASS or CVSNET. The importance of good knowledge engineering will grow as knowledge-based intelligent systems increase in size and complexit

A gross quantitative evaluation of SASS is accomplished by comparing HEARSAY-II' performance with and without SASS. (The modular design of HEARSAY-II permits variable configurations of KSs to be tested.) It must be kept in mind that such an evaluation is affected by the cooperative interactions between KSs, such as the verification by lower-level KSs of words predicted by SASS. Thus the performance difference to be described cannot be credited to SASS alone; in a very real way, the whole of HEARSAY-II is greater than the sum of its parts.

Without the grammatical knowledge provided by SASS (and WOSEQ), HEARSAY-II becomes a bottom-up word recognizer. On average, roughly 80% of the words in an utterance are correctly hypothesized. For every correct word hypothesis, there are 10 or 20 incorrect words. When a correct word is recognized, it is rated lower than an average of four words hypothesized in the same time interval. Thus the highest-rated word sequence spanning the utterance is almost never correct.

With SASS (and WOSEQ), HEARSAY-II performs quite respectably. About 90% of the utterances are recognized correctly or are recognized slightly incorrectly but are correctly interpreted semantically. As explained in the introduction, the appropriat performance criterion for a speech understanding system is how often it understands the intended meaning of an utterance, rather than how often it correctly recognizes all words in the utterance. HEARSAY-II currently operates on average in under 25 times real-time, i.e., a two-second utterance takes about 50 seconds to recognize on a Digital Equipment Corporation KL-10. These results are given for a language with a thousand-word vocabulary and an average branching factor of about ten. Performance on languages with larger branching factors is slower and less accurate.

SASS's performance is reasonably good considering the environment in which it works. The approach taken by SASS seems to be fairly robust with respect to variatio in the rest of HEARSAY-II, but its optimality or sub-optimality within its current environment can only be determined by future research.

11.1 Future Directions

It appears that some utterances not recognized by HEARSAY-II (within preset time and space bounds) could be recognized if SASS could identify partial matches containing "holes," i.e., templates partially instantiated by sequences of words allowing temporal gaps between successive words. For example, if the words TELL, ME, and BEEF are recognized, SASS is unable to detect that the template $LOOKUP1+$TOPIC is well

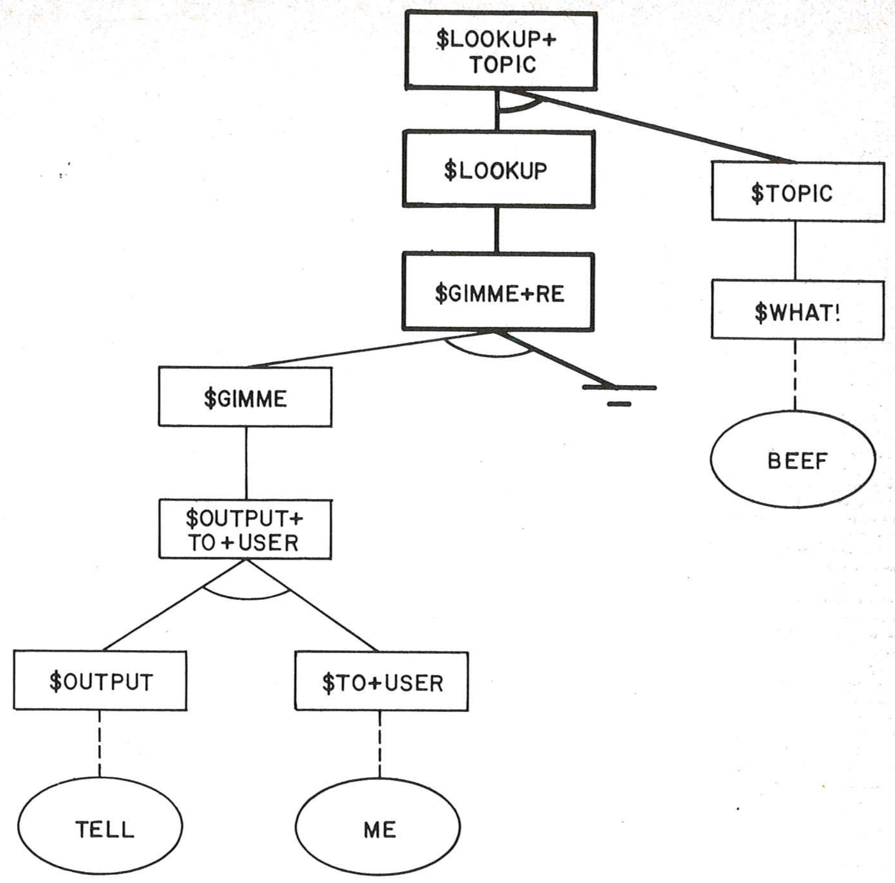

Figure 11.1: Partial match of $LOOKUP+TOPIC with missing middle constituent

partial-matched (see Figure 11.1). The efficient identification of sets of grammatically consistent but non-contiguous words and phrases is a problem under investigation.

Another limitation of SASS is its inability to recognize utterances containing ungrammaticalities such as insertions, deletions, and repetitions. These phenomena are characteristic of spontaneous speech, and must be tolerated by any sophisticated speech understanding system. Most phenomena of this sort arise from the introduction of extraneous information into the utterance. If this information is ignored, the remainder of the utterance is often grammatical. Ignoring an interval of speech means skipping over it, i.e., considering its beginning and end to be temporally adjacent. A mechanism for ignoring intervals is in fact already implemented in SASS, and is used for ignoring intervals of silence. The same mechanism could be used for ignoring extraneous information, given a policy for identifying intervals containing extraneous information. The development of such a policy is another problem under investigation.

12 CONCLUSION

Experience with SASS has provided many interesting lessons, some of which appear to have applications outside the domain of speech understanding.

First, the history of SASS illustrates the value of automatic conversion of static knowledge into practical behavior. The representation of SASS's grammatical knowledge in a nonprocedural form greatly facilitated both the modification of that knowledge and the exploration of different generic behaviors which could be inferred from it.

Second, SASS provides an example of a robust combination of multi-directional problem solving methods. The efficient cooperation of potentially redundant methods is an important problem for "real-world" problem solving systems. When they can be used, strong methods that test some constraint should be preferred to weaker methods that enumerate all constituents potentially satisfying the same constraint.

Third, the effective use of partial solutions is a key to robust problem solving performance. Promising partial solutions should be extended. Rejected partial solutions should be contracted to exploit their possible partial correctness.

Fourth, partial-matching is essential to the process of identifying the best "islands of reliability" (promising partial solutions). However, partial-matching must be highly constrained by some filtering mechanism.

Fifth, effective focus of attention is important in a large inferencing system. The selection of which inference to perform next should be made very carefully to avoid a combinatorial explosion of useless inferences and to accelerate the process of finding the correct solution.

Finally, it should be noted that the success of speech understanding research and of HEARSAY-II in particular indicates maturity and practical potential in the field of artificial intelligence. Such success is an encouraging sign for future research on natural language processing.

REFERENCES

1. Colby, K. M., Faught, B., and Parkinson, R. C. Pattern-matching rules for the recognition of natural language dialogue expression. AIM-234. Stanford AI Laboratory, Stanford.

2. Hayes-Roth, F., and Lesser, V. R. Focus of attention in a distributed logic speech understanding system. *Proceedings of the 1976 I.E.E.E. International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, 1976, 416-420.

3. Hayes-Roth, F., and Mostow, D.J. An automatically compilable recognition network for structured patterns. *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, U.S.S.R., 1975, 246-251.

4. Hayes-Roth, F., and Mostow, D. J. Syntax and semantics in a distributed logic speech understanding system. *Proceedings of the 1976 I.E.E.E. International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, 1975, 421-424.

5. Hayes-Roth, F., and Mostow, D. J. Organization and control of syntactic, semantic, inferential and world knowledge for language understanding. *Proceedings 1976 International Conference on Computational Linguistics*, Ottawa, Canada, 1976.

6. Hayes-Roth, F., Erman, L. D., Fox, M., and Mostow, D. J. Syntactic processing in HEARSAY-II. In Speech Understanding Systems: Summary of Results of the Five-Year Research Effort, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, 1976.

7. Hayes-Roth, F., Fox, M., Gill, G., and Mostow, D. J. Semantics and pragmatics in the Hearsay-II speech understanding system. In Speech Understanding Systems: Summary of Results of the Five-Year Research Effort, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, 1976.

8. Hayes-Roth, F., Gill, G., and Mostow, D. J. Discourse analysis and task performance in the Hearsay-II speech understanding system. In Speech Understanding Systems: Summary of Results of the Five-Year Research Effort, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, 1976.

9. Hayes-Roth, F., Lesser, V. R., Mostow, D. J., and Erman, L. D. Policies for rating hypotheses, halting, and selecting a solution in the Hearsay-II speech understanding system. In Speech Understanding Systems: Summary of Results of the Five-Year Research Effort, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, 1976.

10. Lesser, V. R., Fennell,, R. D., Erman, L. D., and Reddy, D. R. Organization of the Hearsay-II speech understanding system. *I.E.E.E. Transactions on Acoustics, Speech and Signal Processing*, ASSP-23, 1975, 11-33.

11. Lesser, V., Hayes-Roth, F., Birnbaum, M. The word-sequence hypothesizer in Hearsay-II. In Speech Understanding Systems: Summary of Results of the Five-Year Research Effort, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, 1976.

12. Newell, A. Heuristic programming: ill-structured problems. In J. Aronofsky (Ed.), *Progress in Operations Research 3*. New York: Wiley, 1969, 360-414.

13. Newell, A. Artificial Intelligence and the concept of mind. In R. Schank and K. Colby (Eds.), *Computer Models of Thought and Language.* San Francisco: Freeman, 1973, 1-60.

14. Newell, A. Production Systems: models of control structures. In W. C. Chase (Ed.), *Visual Information Processing.* New York: Academic Press, 1973, 463-526.

15. Reddy, D. R. Personal communication, 1976.

16. Smith, A. R. Word hypothesization in the Hearsay-II speech system. *Proceedings I.E.E.E. International Conference on Acoustics, Speech, and Signal Processing,* Philadelphia, 1976, 578-581.

17. Thompson, R. A. Language correction using probabilistic grammars. *I.E.E.E. Transactions on Computers,* C-25, 1976,

18. Walker, D., *et al.* Final report of the SRI-SDC speech understanding system research. Menlo Park: Stanford Research Institute, 1976, in press.

19. Woods, W. A. Transition network diagrams for natural language analysis. Communications of the ACM, 1970, *13*, 591-606.

20. Woods, W. A., *et al.* Final report of the BBN speech understanding system research. Cambridge: Bolt, Beranek, Newman, 1976, in press.

AUTOMATIC SPEECH RECOGNITION OF LARGE VOCABULARIES

via

Dynamic Search Strategies for Information Retrieval from Large Data Bases

G. M. White

Xerox Palo Alto Research Center

3333 Coyote Hill Road

Palo Alto, California 94304/USA

Introduction

This chapter discusses automatic speech recognition techniques for **large vocabularies** of acoustically unambiguous words and short phrases taken from continuous speech or isolated utterances. Techniques that work for the recognition of small vocabularies are not practical for large vocabularies because of the large amount of reference data that must be processed. The key to success in this area centers on **new ways of representing and retrieving knowledge about speech events so that large amounts of speech information may be compactly stored and quickly retrieved.** This is the central problem in speech recognition for large vocabulary systems. Knowledge can be encoded in operations as well as in static memories. In such cases, the retrieval of knowledge amounts to computing a result rather than simply recalling it from from memory. Knowledge about speech can be encoded with compressed signal parameters or in more abstract "rules of operation with symbolic representations". The symbolic representations include phonemic symbols and the rules operation include syntax. The generally accepted approach to the recognition of large vocabularies is to use symbolic representations and appropriate rules of syntax. But there is an advantage to working with signal representations: they contain more information than the symbolic forms derived from them. On the other hand, using signal parameters has the apparent drawback is that it is computationally more expensive to store, retrieve, and process the larger amounts of data required for their use. This drawback can be mitigated by new techniques for retrieving and processing speech data from large data bases. Several new techniques will be presented.

Methods of representing knowledge may be characterized by the extent to which they employ simple memory or operational procedures. Knowledge involving essentially pure memory is called **"eidetic"** knowledge. Eidetic knowledge is based on the retrieval of reference data involving a minimal amount of computation. An example of what we mean by eidetic knowledge is the representation of a speech sound by the storage of a series of numbers representing the amplitude fluctuations as a function of time of the speech sound. Non-eidetic knowledge, or **"procedural"** knowledge as we shall call it, is embodied in sets of operations that enable answers to questions to be computed rather than simply retrieved from memory. The algorithm for long division of two numbers is an example. Given sufficient memory, the answers to questions of division can be answered by table lookup. Clearly a great deal of memory is saved at minor cost in processor time by computing the answers rather than looking them up. The driving force for the use of procedural knowledge is, in general, the reduction in memory requirements achieved through its use. The savings in memory must be paid for by increased computation but the tradeoff is frequently worthwhile.