Organization of the Hearsay II Speech Understanding System

VICTOR R. LESSER, RICHARD D. FENNELL, LEE D. ERMAN, AND D. RAJ REDDY,
MEMBER, IEEE

Abstract—Hearsay II (HSII) is a system currently under development at Carnegie-Mellon University to study the connected speech understanding problem. It is similar to Hearsay I (HSI) in that it is based on the hypothesize-and-test paradigm, using cooperating independent knowledge sources communicating with each other through a global data structure (blackboard). It differs in the sense that many of the limitations and shortcomings of HSI are resolved in HSII.

The main new features of the Hearsay II system structure are: 1) the representation of knowledge as self-activating, asynchronous, parallel processes, 2) the representation of the partial analysis in a generalized three-dimensional network (the dimensions being level of representation (e.g., acoustic, phonetic, phonemic, lexical, syntactic), time, and alternatives) with contextual and structural support connections explicitly specified, 3) a convenient modular structure for incorporating new knowledge into the system at any level, and 4) a system structure suitable for execution on a parallel processing system.

The main task domain under study is the retrieval of daily wire-service news stories upon voice request by the user. The main parametric representations used for this study are 1/3-octave filter-bank and linear-predictive coding (LPC)-derived vocal tract parameters [10], [11]. The acoustic segmentation and labeling procedures are parameter-independent [7]. The acoustic, phonetic, and phonological components [23] are feature-based rewriting rules which transform the segmental units into higher level phonetic units. The vocabulary size for the task is approximately 1200 words. This vocabulary information is used to generate word-level hypotheses from phonetic and surface-phonemic levels based on prosodic (stress) information. The syntax for the task permits simple English-like sentences and is used to generate hypotheses based on the probability of occurrence of that grammatical construct [19]. The semantic model is based on the news items of the day, analysis of the conversation, and the presence of certain content words in the partial analysis. This knowledge is to be represented as a production system. The system is expected to be operational on a 16-processor minicomputer system [3] being built at Carnegie-Mellon University.

This paper deals primarily with the issues of the system organization of the HSII system.

INTRODUCTION

THE HEARSAY II (HSII) speech understanding system is a successor to the Hearsay I (HSI) system [16], [17]. HSII represents, in terms of both its system organization and its speech knowledge, a significant increase in sophistication and generality over HSI. The development of HSII has been based on two years of experience with a running version of HSI, a desire to exploit multiprocessor and network computer architecture for efficient implementation [3], [4], [6], and a desire to handle more complex speech task domains (e.g., larger vocabularies, less restricted grammars, and a more complete set of knowledge sources including prosodics, user models, etc.). While from a conceptual point of view HSII is a natural extension of the framework that HSI posited for a speech understanding system, it differs significantly in its design and in its details of implementation.[1]

THE HEARSAY SYSTEM MODEL

HSI was based on the view that the inherently errorful nature of connected speech processing could be handled only through the efficient use of multiple, diverse sources of knowledge [13], [15]. The major focus of the design of HSI was the development of a framework for representing these diverse sources of knowledge and their cooperation [18]. This framework is the conceptual legacy which forms the basis for the HSII design.

There are four dimensions along which knowledge representation in HSI can be described: 1) function, 2) structure, 3) cooperation, and 4) attention focusing.

The *function* of a knowledge source (KS) in HSI has three aspects. The first is for the KS to know when it has something useful to contribute, the second is to contribute its knowledge through the mechanism of making a *hypothesis* (guess) about some aspect of the speech utterance, and the third is to evaluate the contribution of other knowledge sources, i.e., to verify and reorder (or reject) the hypotheses made by other knowledge sources. Each of these aspects of a KS is carried out with respect to a particular context, the context being some subset of the previously generated hypotheses. Thus, new knowledge is built upon the educated guesses made at some previous time by other knowledge sources.

The *structure* of each knowledge source in HSI is specified so that it is independent and separable from all other KS's in the system. This permits the easy addition of new

Manuscript received March 29, 1974; revised August 15, 1974. This research was supported in part by the Defense Advanced Research Projects Agency under Contract F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research. This paper was presented at the IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pa., April 15–19, 1974.

The authors are with the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa. 15213.

[1] Several other organizations for speech understanding systems have been implemented over the past few years. Included among the more interesting ones are the SDC system [2], [20], the Bolt Beranek and Newman (BBN) SPEECHLIS system [21], [24], the DRAGON system [1], and the IBM system [9].

types of KS's and replacement of KS's with alternative versions of those KS's. Thus, the system structure can be easily adapted to new speech task domains which have KS's specific to that domain, and the contribution of a particular KS to the total recognition effort can be more easily evaluated.

The choice of a framework for *cooperation* among KS's is intimately interwoven with the function and structure of knowledge in HSI. The mechanism for KS cooperation involves *hypothesizing* and *testing* (creating and evaluating) hypotheses in a *global data base* (blackboard). The generation and modification of globally accessible hypotheses thus becomes the primary means of communication between diverse KS's. This mechanism of cooperation allows a KS to contribute knowledge without being aware of which other KS's will use its knowledge or which KS contributed the knowledge that it used. Thus, each KS can be made independent and separable.

The global data base that KS's use for cooperation contains many possible interpretations of the speech data. Each of these interpretations represents a "limited" context in which a KS can possibly contribute information by proposing or validating hypotheses. *Attention focusing* of a KS involves choosing which of these limited contexts it will operate in and for how much processing time. The attention focusing strategy is decoupled from the functions of individual KS's. Thus, the decision of whether a KS can contribute in a particular context is local to the KS, while the assignment of that KS to one of the many contexts on which it can possibly operate is made more globally. This decoupling of focusing strategy from knowledge acquisition, together with the decoupling of the data environment (global data base) from control flow (KS invocation) and the limited context in which a KS operates, permits a quick refocusing of attention of KS's. The ability to refocus quickly is very important in a speech understanding system, because the errorful nature of the speech data and its processing leads to many potential interpretations of the speech. Thus, as soon as possible after an interpretation no longer seems the most promising, the activity of the system should be refocused to the new most promising interpretation.

OVERVIEW OF HSI

The following is a brief description of the HSI implementation for this model of knowledge source representation and cooperation. This description will then be used to contrast the differences of implementation philosophy between HSI and HSII. (More complete descriptions of HSI are contained in [5], [12], [16], [17].)

HSI Implementation Overview

The global data base of HSI consists of partial sentence hypotheses, each being a sequence of words with nonoverlapping time locations in the utterance. It is a *partial* sentence hypothesis because not all of the utterance need be described by the given sequence of words. In particular, gaps in the knowledge of the utterance are designated by "filler" words. The partial sentence hypotheses also contain confidence ratings for each word hypothesis and a composite rating for the overall sequence of words. A sentence hypothesis is the focal point that is used to invoke a knowledge source. The sentence hypothesis also contains the accumulation of all information that any KS has contributed to that hypothesis.

KS's are invoked in a lockstep sequence consisting of three phases: *poll*, *hypothesize*, and *test*. At each phase, all KS's are invoked for that phase, and the next phase does not commence until all KS's have completed the current one. The poll phase involves determining which KS's have something to contribute to the sentence hypothesis which is currently being focused upon; polling also determines how confident each KS is about its proposed contributions. The hypothesize phase consists of invoking the KS showing the most confidence about its proposed contribution of knowledge. This KS then hypothesizes a set of possible words (option words) for some (one) "filler" word in the speech utterance. The testing phase consists of each KS evaluating (verifying) the possible option words with respect to the given context. After all KS's have completed their verifications, the option words which seem most likely, based on the combined ratings of all the KS's, are then used to construct new partial sentence hypotheses. The global data base is then re-evaluated to find the most promising sentence hypothesis; this hypothesis then becomes the focal point for the next hypothesize-and-test cycle.

HSI Performance

The HSI system first demonstrated live, connected-speech recognition publicly in June 1972. Since that time, about three man-years have been spent in improving its performance. The system has been tested on a set of 144 connected speech utterances, containing 676 word tokens, spoken by five speakers, and consisting of four tasks, with vocabularies ranging from 28 to 76 words. On the average, the system locates and correctly identifies about 93 percent of the words, using all of its KS's. Without the use of the Semantics KS, the accuracy decreases to 70 percent. It decreases further to about 39 percent when neither Syntax nor Semantics are used. A more complete performance summary may be found in the Appendix.

HSI Design Limitations

There are four major design decisions in the HSI implementation of knowledge representation and cooperation which make it difficult to apply HSI to more complex speech tasks or multiprocessor environments.

The *first*, and most important, of these limiting decisions concerns the use of the hypothesize-and-test paradigm. As implemented in HSI, the paradigm is exploited only at the word level. The implication of hypothesizing and testing at only the word level is that the knowledge representation is uniform only with respect to cooperation at that level. That is, the information content of any

element in the global data base is limited to a description at the word level. The addition of nonword level KS's (i.e., KS's cooperating via either subword levels, such as syllables or phones, or via supraword levels, such as phrases or concepts) thus becomes cumbersome because this knowledge must somehow be related to hypothesizing and testing at the word level. This approach to nonword level KS's makes it difficult to add nonword knowledge and to evaluate the contribution of this knowledge. In addition, the inability to share nonword level information among KS's causes such information to be recomputed by each KS that needs it.

Second, HSI constrains the hypothesize-and-test paradigm to operate in a lockstep control sequence. The effect of this decision is to limit parallelism, because the time required to complete a hypothesize-and-test cycle is the maximum time required by any single hypothesizer KS plus the maximum time required by any single verifier (testing) KS. Another disadvantage of this control scheme is that it increases the time it takes the system to refocus attention, because there is no provision for any communication of partial results among KS's. Thus, for example, a rejection of a particular option word by a KS will not be noticed until all the KS's have tested all the option words.

The *third* weakness in the HSI implementation concerns the structure of the global data base: there is no provision for specifying relationships among alternative sentence hypotheses. The absence of relational structures among hypotheses has the effect of increasing the overall computation time and increasing the time to refocus attention, because the information gained by working on one hypothesis cannot be shared by propagating it to other relevant hypotheses.

The *fourth* limiting design decision relates to how a global problem-solving strategy (policy) is implemented in HSI: policy decisions, such as those involving attention focusing, are centralized (in a "Recognition Overlord"), and there is no coherent structure for the policy algorithms. The effect of having no explicit system structure for implementing policy decisions makes it very awkward to add or delete new policy algorithms and difficult to analyze the effectiveness of a policy and its interaction with other policies.

OVERVIEW OF HSII

Experience with HSI (as described above) has led to several important observations about a more general, uniform, and natural structure for representing and operating on the (dynamic) state of the utterance recognition.[2]

1) The information contained in hypotheses at different levels of knowledge representation may be encoded in essentially identical internal structures, except for the primitive unit of information held in an hypothesis. This structural homogeneity in the global data base allows the actions of hypothesizing and testing at these various levels to be treated in a uniform manner.

2) The different types of knowledge (and their relationships) present in speech may be naturally represented in a single, uniform data structure. This data structure is three-dimensional: one dimension represents information levels (e.g., phrasal, lexical, phonetic), the second represents speech time, and the third dimension contains alternative (competing) hypotheses at a particular level and time. These three dimensions form a convenient addressing structure for locating hypotheses.

3) There is a conceptually simple and uniform way of dynamically relating hypotheses at one level of knowledge to alternative hypotheses at that level and to hypotheses at other knowledge levels in the structure. The resulting structure is an AND/OR graph with modifications which provide for temporal relationships and selective dependency relationships.[3]

System Structure

The main goal of the HSII design is to extend the concepts developed in HSI for the representation and cooperation of knowledge at the word level to *all levels of knowledge* needed in a speech understanding system, based on the preceding observations.

The generalization of the hypothesize-and-test paradigm to all levels of speech knowledge implies the need for a mechanism for transferring information among levels. This mechanism is already embodied in the hypothesize-and-test paradigm; that is, one can characterize two types of hypothesization a knowledge source might be called upon to perform: horizontal and vertical hypothesization. A hypothesization is *horizontal* when a KS uses contextual information at a given knowledge level to predict new hypotheses at the same level (e.g., the hypothesization that the word "night" might follow the sequence of words "day"–"and"), whereas a hypothesization is *vertical* when a KS uses information at one level in the data base to predict new hypotheses at a different level (e.g., the generation of a hypothesis that a [T] occurred when a segment of silence is followed by a segment of aspiration).

The HSII implementation of the hypothesize-and-test paradigm has also resulted in a generalization of the lockstep control scheme for KS sequencing employed by HSI. HSII relaxes the constraints on the hypothesize-and-test paradigm and allows the KS processes to run in an asynchronous, data-directed manner. A KS is instantiated as a *KS process* whenever the data base exhibits characteristics which satisfy a "precondition" of the KS. A *precondition* of a KS is a description of some partial state of the data base which defines when and where the KS can contribute its knowledge by modifying the data base. Such a modification might be adding new hypotheses proposed by the KS (at the information level appropriate for that KS) or verifying (criticizing) hypotheses which already exist.

[2] The meaning of these observations will be made more clear by the further descriptions that follow.

[3] This latter feature refers to "connection matrices" and is described below in more detail.

The modifications made by any given KS process are expected to trigger further KS's by creating new conditions in the data base to which those KS's respond. The structure of a hypothesis has been so designed as to allow the preconditions of most KS's to be sensitive to a single, simple change in some hypothesis (such as the changing of a rating or the creation of a structural link). Through this data-directed interpretation of the hypothesize-and-test paradigm, HSII KS's exhibit a high degree of asynchrony and potential parallelism. A side-effect of this more general control scheme for HSII is that the overall problem-solving strategy need not be centralized and implemented as a monolithic overlord, but rather can be implemented as *policy modules* which operate in precisely the same manner as KS's.

The three-dimensional data base, augmented by the AND/OR structural relationships specified over that data base, permits information generated by one KS to be: 1) retained for use by other KS's, and 2) quickly propagated to other relevant parts of the data base. This retention and propagation provide two important features for solving a complex problem in which errors are highly likely. First, quick refocusing can occur when a particular path no longer appears promising. Second, "selective" backtracking may be used; i.e., when a KS finds that it has made an incorrect decision, it does not have to eliminate all information generated subsequent to that decision, but rather only that subset which depends on the incorrect decision. In this way, information generated by one knowledge source is retained and is usable by itself and other KS's in other relevant contexts.

Summarizing, HSII is based on the views: 1) that the state of the recognition can be represented in a uniform, multilevel data base, and 2) that speech knowledge can be characterized in a natural manner by describing many small KS's. These KS's react to certain states of the data base (via their preconditions) and, once instantiated as KS processes, provide their own changes to the data base which contribute to the progress of the recognition. The hypothesize-and-test paradigm, when stated in sufficiently nonrestrictive (parallel) terms, serves to describe the general interactions among these KS's. In particular, changes made by one or more KS processes may trigger other KS's to react to these changes by validating (testing) them or hypothesizing further changes. The intent of HSII is to provide a framework within which to explore various configurations of information levels, KS's, and global strategies.[4]

From a more general point of view, the goal of HSII is to provide a multiprocess-oriented software architecture to serve as a basis for systems of cooperating (but independent and asynchronous) data-directed KS processes. The purpose of such a structure is to achieve effective parallel search over a general artificial intelligence problem-solving graph, employing the hypothesize-and-test paradigm to generate the search graph and using a uniform, interconnected, multilevel global data base as the primary means of interprocess communication.

[4] It is interesting to note that this generalized form of hypothesize-and-test leads to a system organization with some characteristics similar to QA4 [22] and PLANNER [8]. In particular, there are strong similarities in the data-directed sequencing of processes.

HSII SYSTEM DESIGN AND IMPLEMENTATION

One can derive from the description of the desired HSII recognition process given above several basic components of the required system structure. First, a sufficiently general *structured global data base* is needed, through which the KS's may communicate by inserting hypotheses and by inspecting and modifying the hypotheses placed there by other KS's. Second, some means for describing the various KS's and their internal processing capabilities is required. Third, in order to have knowledge sources activated in a data-directed manner, a method is required by which a set of *preconditions* may be specified and associated with each KS. Fourth, in order to detect the satisfaction of these preconditions and in order to allow KS's to locate parts of the data base in which they are interested, two mechanisms are needed: 1) a *monitoring mechanism* to record where in the data base changes have occurred and the nature of those changes, and 2) an *associative retrieval mechanism* for accessing parts of the data base which conform to particular patterns specified by *matching prototypes*.

Elements of the System Structure

The following sections outline the HSII implementation of the various basic system components.

Global Data Base: The design of HSII is centered around a global data base (blackboard) which is accessible to all KS processes. The global data base is structured as a uniform, multilevel, interconnected data structure.

Each level in the data base contains a (potentially complete) representation of the utterance; the levels are differentiated by the units that make up the representation, e.g., phrases, words, phonemes. The system structure of HSII does not prespecify what the levels in the global data structure are to be. A particular configuration, called HSII-C0 (Configuration Zero), is being implemented as the first test of the HSII structure. Fig. 1 shows a schematic of the levels of HSII-C0. A more detailed description and justification for parts of this particular configuration can be found in [23]. This configuration will be used as the basis for examples to illustrate various aspects of the HSII system.

Parametric Level: The parametric level holds the most basic representation of the utterance that the system has; it is the only direct input to the machine about the acoustic signal. Several different sets of parameters are being used in HSII-C0 interchangeably: 1/3-octave filter-band energies measured every 10 ms, LPC-derived vocal-tract parameters [10], and wide-band energies and zero-crossing counts.

Segmental Level: This level represents the utterance as labeled acoustic segments. Although the set of labels may

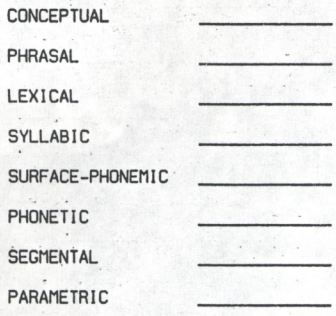

Fig. 1. Levels in HSII-C0.

be phonetic-like, the level is not intended to be phonetic—the segmentation and labeling reflect acoustic manifestation and do not, for example, attempt to compensate for the context of the segments or attempt to combine acoustically dissimilar segments into (phonetic) units.

As with all levels, any particular portion of the utterance may be represented by more than one competing hypothesis (i.e., multiple segmentations and labelings may co-exist).

Phonetic Level: At this level, the utterance is represented by a phonetic description. This is a *broad* phonetic description in that the size (duration) of the units is on the order of the "size" of phonemes; it is a *fine* phonetic description to the extent that each element is labeled with a fairly detailed allophonic classification (e.g., "stressed, nasalized [I]").

Surface-Phonemic Level: This level, named by seemingly contradicting terms, represents the utterance by phoneme-like units, with the addition of modifiers such as stress and boundary (word, morpheme, syllable) markings.

Syllabic Level: The unit of representation here is the syllable.

Lexical Level: The unit of information at this level is the word. (Note again that at any level competing representations can be accommodated.)

Phrasal Level: Syntactic elements appear at this level. In fact, since a level may contain arbitrarily many "sublevels" of elements structured as a modified AND/OR graph, traditional kinds of syntactic trees can be directly represented here.

Conceptual Level: The units at this level are "concepts." As with the phrasal level, it may be appropriate to use the graph structure of the data base to indicate relationships among different concepts.

The basic unit in the data structure is a *node*; a node represents the *hypothesis* that a particular element exists in the utterance. For example, a hypothesis at the phonetic level may be labeled as "T." Besides containing the hypothesis element name, a node holds several other kinds of information, including: 1) a correlation with a particular *time period* in the speech utterance,[5] 2) *scheduling parameters* (validity ratings, attention-focusing factors, measures of computing effort expended, etc.), and 3) *connection information* which relates the node to other nodes in the data base.

Structural relationships between nodes (hypotheses) are represented through the use of *links*; links provide a means for specifying contextual abstractions about the relationships of hypotheses. A link is an element in the data structure which associates two nodes as an ordered pair; one of the nodes is termed the *upper hypothesis*, and the other is called the *lower hypothesis*. The lower hypothesis is said to *support* the upper hypothesis; the upper hypothesis is called a *use* of the lower one. There are several types of links; in general, if a node serves as the upper hypothesis for more than one link, all of those links must be of the same type. Thus, one can talk of the "type of the hypothesis," which is the same as the type of all of its lower links. The two most important structural relationship types are SEQUENCE and OPTION.

A SEQUENCE node is a hypothesis that is supported by a (timewise) sequential set of hypotheses at a lower level (or sublevel—see below). For example, Fig. 2(a) shows a hypothesis of "will" at the lexical level supported by the (time-)ordered contiguous sequence of "W," "IH," and "L" at the surface-phonemic level. The interpretation of a SEQUENCE node is that all of the lower hypotheses must be valid in order to support the upper hypothesis.

An OPTION node is a hypothesis that has alternative supports from two or more hypotheses, each of which covers essentially the same time period. For example, Fig. 2(b) shows the hypothesis of "noun" at the phrasal level as being supported by any of "boy," "toy," or "tie," all of which are competing word hypotheses covering (approximately) the same time area.[6]

Fig. 3 is a example of a larger fragment of the global data structure. The level of a hypothesis is indicated by its vertical position, the names of the levels are given on the left. Time location is approximately indicated by horizontal placement, but duration is only very roughly indicated (e.g., the boxes surrounding the two hypotheses at the phrasal level should be much wider). Alternatives are indicated by proximity; for example, "will" and "would" are word hypotheses covering the same time span. Likewise, "question" and "modal-question," "you1" and "you2," and "J" and "Y" all represent pairs of alternatives.

This example illustrates several features of the data structure.

The hypothesis "you," at the lexical level, has two alternative phonemic "spellings" indicated; the hypothe-

[5] This time period is specified with an explicit estimation of fuzziness, even to the extent of spanning the entire utterance. As an extreme example, if a sequence of "phonemes" is being hypothesized from a "word" at the lexical level, the time boundaries of the phonemes are specified as "fuzzy" if information regarding their actual locations is not yet available.

[6] In addition to SEQUENCE and OPTION, there are two kinds of structural relationships which are generalizations of them: An AND node is similar to a SEQUENCE node except that there is no implication of any time sequentiality among the supports—they may overlap or be disjoint. An OR node is similar to an OPTION node in that the supports represent alternatives, but (as with the AND node) there is no time requirement.

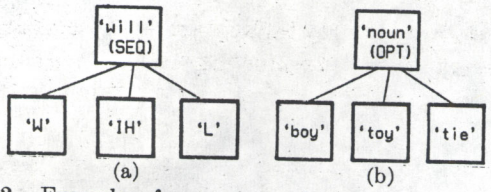

Fig. 2. Examples of SEQUENCE and OPTION relationships.

ses labeled "you1" and "you2" are nodes created, also at the lexical level, to hold those alternatives. In general, such *sublevels* may be created arbitrarily.

The link between "you1" and "D" is a special kind of SEQUENCE link (indicated here by a dashed line) called a CONTEXT link; a CONTEXT link indicates that the lower hypothesis supports the upper one and is contiguous to its brother links, but it is not "part of" the upper hypothesis in the sense that it is not within the time interval of the upper hypothesis—rather, it supplies a context for its brother(s). In this case, one may "read" the structure as stating " "you1" is composed of "J" followed by "AX" (schwa) in the context of the preceding "D." " (This reflects the phonological rule that "would you" is often spoken as "would-ja.") Thus, a CONTEXT link allows important contextual relationships to be represented without violating the implicit time assumptions about SEQUENCE nodes.

Whereas the phonemic spelling of the word "you" held by "you1" includes a contextual constraint, the "you2" option does not have this constraint. However, "you1" and "you2" are such similar hypotheses that there is strong reason for wanting to retain them as alternative options under "you" (as indicated in Fig. 3), rather than representing them unconnectedly. In general, the problem is that the use of a hypothesis implies certain contextual assumptions about its environment, while the support of a hypothesis may itself be predicated on a particular set of contextual assumptions. A mechanism, called a *connection matrix*, exists in the data structure to represent this kind of relationship by specifying, for an OPTION hypothesis, which of its alternative supports are applicable ("connected") to which of its uses. In this example, the connection matrix of "you" (symbolized in Fig. 3 by the two-dimensional binary matrix in the node) specifies that support "you1" is relevant to use "question" (but not to "modal-question") and that support "you2" is relevant to both uses. The use of a connection matrix allows the efforts of preceding KS decisions to be accumulated for future use and modification without necessitating contextual duplication of parts of the data base. Thus, much of the duplication of effort due to the backtracking mode of HSI is avoided in HSII.

Besides showing structural relationships (i.e., that one unit is composed of several other units), a link is a statement about the degree to which one hypothesis implies (i.e., gives evidence for the existence of) another hypothesis. The strength of the implication is held as information on the link. The sense of the implication may be negative; that is, a link may indicate that one hypothesis is evidence

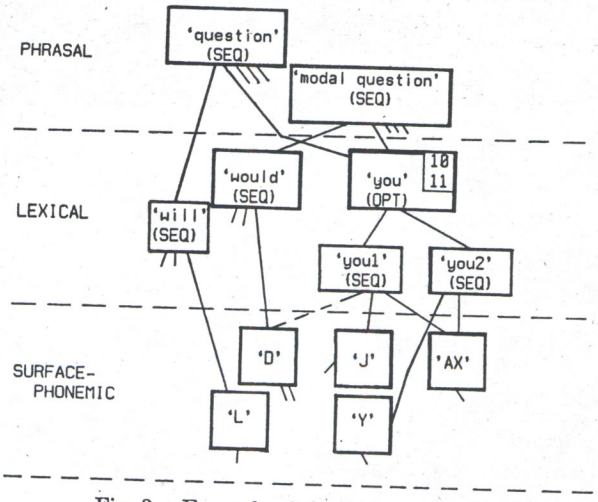

Fig. 3. Example of the data structure.

for the invalidity of another. Finally, this statement of implication is bidirectional; the existence of the upper hypothesis may give credence to the existence of the lower hypothesis and vice versa.

The nature of the implications represented by the links provides a uniform basis for propagating changes made in one part of the data structure to other relevant parts without necessarily requiring the intervention of particular KS's at each step. Considering the example of Fig. 3, assume that the validity of the hypothesis labeled "J" is modified by some KS (presumably operating at the phonetic level) and becomes very low. One possible scenario for rippling this change through the data base is the following.

First, the estimated validity of "you1" is reduced, because "J" is a lower hypothesis of "you1."

This, in turn, may cause the rating of "you" to be reduced.

The connection matrix at "you" specifies that "you1" is not relevant to "modal-question," so the latter hypothesis is not affected by the change in rating of the former. Notice that the existence of the connection matrix allows this decision to be made locally in the data structure, without having to search back down to the "D" and "J."

"Question," however, is supported by "you1" (through the connection matrix at "you"), so its rating is affected.

Further propagations can continue to occur, perhaps down the other SEQUENCE links under "question" and "you1."

KS Specification: A knowledge source is specified in three parts: 1) the conditions under which it is to be activated (in terms of the data base conditions in which it is interested, as described in the section on "preconditions" below), 2) the kinds of changes it makes to the global data base, and 3) a procedural statement (program) of the algorithm which accomplishes those changes. A KS is thus defined as possessing some processing capability which is able to solve some subproblem, given appropriate circumstances for its activation.

The decomposition of the overall recognition task into

various knowledge sources is regarded as being a natural decomposition. That is, the units of the decomposition represent those pieces of knowledge which can be distinguished and recognized as being somehow naturally independent.[7] Given a sufficient set of such KS's (that is, a set that provides enough overall connectivity among the various levels of the data base that a final recognition can be attained), the collection is called the "overall recognition system." Such a scheme of "inverse decomposition" (or, composition) seems very natural for many problem-solving tasks, and it fits well into the hypothesize-and-test approach to problem-solving. As long as a sufficient "covering set" for the data base connections is maintained, one can freely add new KS's, or replace or delete old ones. Each KS is in some sense self-contained, but each is expected to cooperate with the other KS's that happen to be present in the system at that time.

As examples of KS's, Fig. 4 shows the first set of processes implemented for HSII-C0. The levels are indicated as horizontal lines in the figure and are labeled at the left. The KS's are indicated by arcs connecting levels; the starting point(s) of an arc indicates the level(s) of major "input" for the KS, and the end point indicates the "output" level where the KS's major actions occur. In general, the action of most of these particular KS's is to create links between hypotheses on its input level(s) and: 1) existing hypotheses on its output level, if appropriate ones are already there, or 2) hypotheses that it creates on its output level.

The *Segmenter-Classifier* KS uses the description of the speech signal to produce a labeled acoustic segmentation. (See [7] for a description of the algorithm being used.) For any portion of the utterance, several possible alternative segmentations and labels may be produced.

The *Phone Synthesizer* uses labeled acoustic segments to generate elements at the phonetic level. This procedure is sometimes a fairly direct renaming of an hypothesis at the segmental level, perhaps using the context of adjacent segments. In other cases, phone synthesis requires the combining of several segments (e.g., the generation of [t] from a segment of silence followed by a segment of aspiration) or the insertion of phones not indicated directly by the segmentation (e.g., hypothesizing the existence of an [l] if a vowel seems velarized and there is no [l] in the neighborhood). This KS is triggered whenever a new hypothesis is created at the segmental level.

The *Word Candidate Generator* uses phonetic information (primarily just at stressed locations and other areas of high phonetic reliability) to generate word hypotheses. This is accomplished in a two-stage process, with a stop at the syllabic level, from which lexical retrieval is more effective.

[7] The approach taken in knowledge source decomposition is not an attempt to characterize somehow the overall recognition process and then apply some sort of traffic flow analysis to its internal workings in order to decompose the total process into minimally interacting KS's. Rather, KS's are defined by starting with some intuitive notion about the various pieces of knowledge which could be incorporated in a useful way to help achieve a solution.

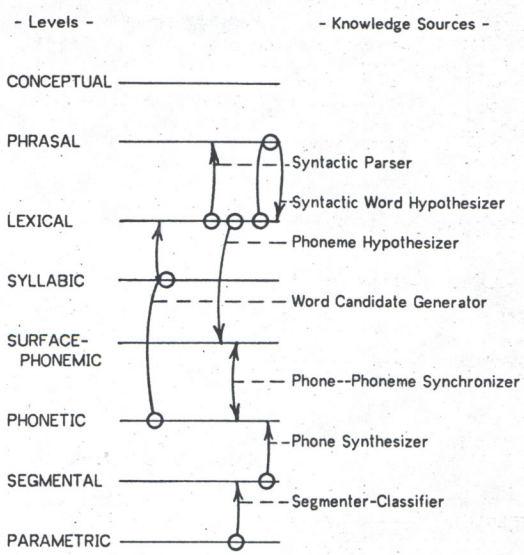

Fig. 4. First KS's for HSII-C0.

The *Syntactic Word Hypothesizer* uses knowledge at the phrasal level to predict possible new words at the lexical level which are adjacent (left or right) to words previously generated at the lexical level. ([19] contains a description of the probabilistic syntax method being used here.) This KS is activated at the beginning of an utterance recognition attempt and, subsequently, whenever a new word is created at the lexical level.

The *Phoneme Hypothesizer* KS is activated whenever a word hypothesis is created (at the lexical level) which is not yet supported by hypotheses at the surface-phonemic level. Its action is to create one or more sequences at the surface-phonemic level which represent alternative pronunciations of the word. (These pronunciations are currently prespecified as entries in a dictionary.)

The *Phone–Phoneme Synchronizer* is triggered whenever a hypothesis is created at either the phonetic or the surface-phonemic level. This KS attempts to link up the new hypothesis with hypotheses at the other level. This linking may be many-to-one in either direction.

The *Syntactic Parser* uses a syntactic definition of the input language to determine if a complete sentence may be assembled from words at the lexical level.

Fig. 5 shows the initial HSII-C0 KS's of Fig. 4 augmented with four additional ones which are being implemented or are planned.

The *Semantic Word Hypothesizer* uses semantic and pragmatic information about the task (news retrieval, in this case) to predict words at the lexical level.

The *Phonological Rule Applier* rewrites sequences at the surface-phonemic level. This KS is used: 1) to augment the dictionary lookup of the Phoneme Hypothesizer, and 2) to handle word boundary conditions that can be predicted by rule.

The primary duties of the *Segment-Phone Synchronizer* and the *Parameter-Segment Synchronizer* are similar: to recover from mistakes make by the (bottom-up) actions of the Phone Synthesizer and Segmenter–Classifier, re-

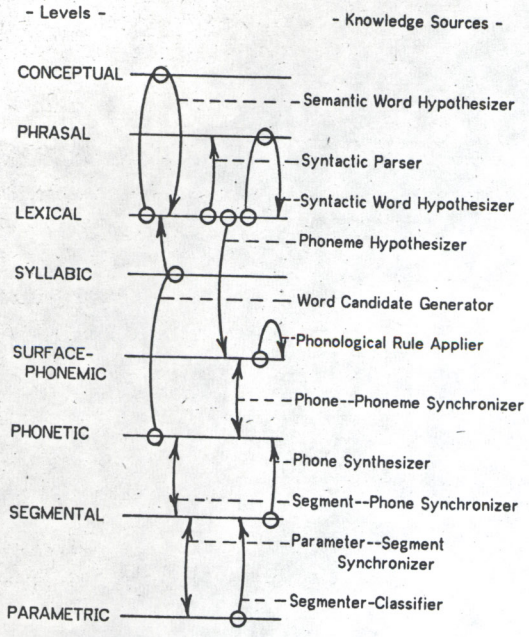

Fig. 5. Augmented KS set for HSII-C0.

spectively, by allowing feedback from the higher to the lower level.

The introduction of these knowledge sources indicates the modularity and extendability of the system in terms of both KS's and levels. In particular, notice that even though the purpose of some KS is stated as "correcting errors produced by other KS's," each KS is independent of the others. Yet additional KS's will be added to the configuration as the need for them is seen and as ideas for their implementation are developed.

In addition to the KS modules described above, all of which embody speech knowledge, several policy modules exist. These modules, which interface to the system in a manner identical to the speech modules, execute policy decisions, e.g., propagation of ratings, focusing of attention, and scheduling of other modules.

Matching-Prototypes and Associative Retrieval: The synchronous processing activity in HSII is primarily data-directed; this implies the need for some mechanism whereby one can retrieve parts of the global data base in an associative manner. HSII provides primitives for associatively searching the data base for hypotheses satisfying specified conditions (e.g., finding all hypotheses at the phonetic level which contain a vowel within a certain time range). The search condition is specified by a *matching prototype*, which is a partial specification of the components of a hypothesis. This partial specification permits a component to be characterized by: 1) a set of desired values, or 2) a DON'T-CARE condition, or 3) values of components of a hypothesis previously derived by matching another prototype. A matching prototype can be compared against a set of hypotheses. Those hypotheses whose component values match those specified by the matching prototype are returned as the result of the search. Associative retrieval of structural relationships among hypotheses is also provided by several primitives. More complex retrievals can be accomplished by combining the retrieval primitives in appropriate ways.

Preconditions and Change Sets: Associated with every KS is a specification of the data base conditions required for the instantiation of that KS. Such specifications, called *preconditions*, conceptually form an AND/OR tree composed of matching prototypes and structural relationships which, when applied to the data base in an associative manner, detect the regions of the data base in which the KS is interested (if the precondition is capable of being satisfied at that time). Alternatively, one might think of the precondition specification as a procedure, involving matching prototypes and structural relationships, which effectively evaluates a conceptual AND/OR tree. This procedure may contain arbitrarily complex decisions (based on current and past modifications to the data base) resulting in the activation of desired KS's within the chosen contexts. The context corresponding to the discovered data base region which satisfies some KS's precondition is used as an initial context in which to instantiate that KS as a new process. If there are multiple regions in the data base that satisfy the specified conditions, the KS can be separately instantiated for each context, or once with a list of all such contexts.

Whenever any KS performs a modification to the global data base, the essence of the modification is preserved in a *change set* appropriate to the change made (e.g., one change set records rating changes, while another records time range changes). Change sets serve to categorize data base modifications (events) and are thus useful in precondition evaluation since they limit the areas in the data base that need to be examined in detail. As currently implemented, precondition evaluators exist as a set of procedures which monitor changes in the data base (i.e., they monitor additions to those change sets in which they are interested). These precondition procedures are themselves data-directed in that they are applied whenever sufficient changes have been made in the global data base. In effect, the precondition procedures themselves have preconditions, albeit of a much simpler form than those possible for KS's. For example, a precondition procedure may specify that it should be invoked (by a system precondition invoker) whenever changes occur to two adjacent hypotheses at the word level or whenever support is added to the phrasal level. By using the (coarse) classifications afforded by change sets, the system avoids most unnecessary data base examinations by the precondition procedures. The major point to be made is that the scheme of precondition evaluation is *event driven*, being based on the occurrence of changes in the global data base. In particular, precondition evaluators are not involved in a form of busy waiting in which they are constantly looking for something that is not yet there.

Once invoked, a precondition procedure uses sequences of associative retrievals and structural matches on the data base in an attempt to establish a context satisfying the preconditions of one or more of "its" KS's; any given

precondition procedure may be responsible for instantiating several (usually related) KS's. Notice that the data-directed nature of precondition evaluation and KS instantiation is linked closely to the primitive functions that are able to modify the data base, for it is only at points of modification that a precondition that was unsatisfied before may become satisfied. Hence, data base modification routines have the responsibility (although perhaps indirectly) of activating the precondition evaluation mechanism.[8]

Multiprocessing Considerations: Some Problems and Their Solutions

A primary goal in the design of HSII is to exploit the potential parallelism of the Hearsay system model as fully as possible. Several issues associated with the introduction of parallel KS processes into HSII will be described and their current solutions outlined.

Local Contexts: A precondition evaluator (process) is invoked based on the occurrence of certain changes which have taken place in the global data base since the last time the evaluator was invoked; these changes, together with the state of the relevant parts of the global data base at the instant at which the precondition evaluator is invoked, form a *local context* within which the evaluator operates. Conceptually, at the instant of its invocation, the precondition evaluator takes a snapshot of the global data base and saves the substance of the changes that have occurred to that moment, thereby preserving its local context.

The necessity of preserving this local context exists for several reasons: 1) HSII consists of asynchronous processes sharing a common global data base which contains only the most current data (that is, no history of data modification is preserved in the global data base), 2) since the precondition evaluators are also executed asynchronously, each evaluator is interested only in changes in the data base which have occurred since the last time that particular evaluator was executed (that is, the relevant set of changes for a particular precondition evaluator is time-dependent on the last execution of that evaluator), and 3) further modifications to the global data base which are of interest to a given KS process may occur between the time of that KS process's instantiation and the time of its completion (in particular, such modifications and their relationship to data base values which existed at the time of the instantiation of the KS process may influence the computation of that KS process). Hence, the problem of creation of local contexts exists, as do the associated problems of signaling a KS process

[8] One might think of HSII as a production system [14] which is executed asynchronously. The preconditions correspond to the left-hand sides (conditions) of productions, and the knowledge sources correspond to the right-hand sides (actions) of the productions. Conceptually, these left-hand sides are evaluated continuously. When a precondition is satisfied, an instantiation of the corresponding right-hand side of its production is created; this instantiation is executed at some arbitrary subsequent time (perhaps subject to instantiation scheduling constraints).

```
T1:  start precondition evaluator PRE
       (triggered by data base changes)
T2:  PRE instantiates a knowledge-source process KS
T3:  end PRE

T4:  start KS
T5:  after KS revalidation of precondition
     <computation>
T6:  KS modifies global data base
     <computation>
T7:  KS modifies global data base again
T8:  end KS
```

Fig. 6. Timing sequence of a precondition evaluator and KS.

when its local context is no longer valid and of updating these contexts as further changes occur in the global data base.

Consider the time sequence of events shown in Fig. 6. The precondition evaluator PRE is activated to respond to changes occurring in the global data base. PRE should execute in the context of changes existing at time T1 (since that context contains the changes which caused PRE to be activated). KS is instantiated (readied for running) at T2 due to further conditions PRE discovered about the change context of T1. Hence, PRE should pass the context of T1 as the initial environment in which to run KS.

By time T4, when KS actually starts to execute, other changes could have occurred in the global data base due to the actions of other KS processes. So KS should examine these new updating changes (those occurring between T1 and T4) and revalidate its precondition, if necessary. After revalidation, KS assumes the updated context of T5, and it proceeds to base its computation on the context of changes as of T5.

When KS wishes to perform an actual update of elements of the global data base at T6, it must examine the changes to the global data base that have occurred between T5 and T6 to see if any other KS processes may have violated KS's preconditions, thereby invalidating its computations. Having performed this revalidation and any data base updating, KS should update its context to reflect changes up to T6 for use in its further computation. At T7, KS must look for further possible invalidations to its most recent computations, due to possible changes in the global data base by other KS processes during the time period T6 to T7. When KS (which is an instantiation of some KS) completes its actions at T8, its local context may be deleted.

Changes occurring between instantiations of a KS are accumulated in the local contexts of the precondition evaluators and may become part of the local context of a future instantiation of a KS. Thus, the precondition evaluators are always collecting data base changes (since these evaluators are permanently instantiated), while individual KS instantiations accumulate data base changes only during their transient existence.

In practice, to create local contexts one need only save the contents of *changes* which occur in the global data base (thereby allowing the global data base to contain only the very latest values). In particular, no massive copy operations involving the global data base are required.

Thus, for each data base event caused by a modification primitive, the associated change[9] is distributed (copied) into the local contexts (which can now be characterized as *local change sets*,[10] referring to the previous discussion on change sets) of all KS processes and precondition evaluators who care. Notice that not every KS process and precondition evaluator cares about every change that takes place. For example, a fricative detector will not care about a data base change associated with the grouping of several words to form a syntactic phrase, but it is interested in the hypothesization of a word whose phonemic spelling contains a fricative.

In order for a KS (or precondition evaluator) process to be notified of changes which occur to particular fields of particular nodes which are in its local context, those fields need to be *tagged* with an alias (called a *tagid*) belonging to that KS instantiation. Then, whenever a modification is made to the global data base, a *message* signaling the change is sent to all who have tagged the field being changed. In addition, the contents of the change are also distributed to the local contexts of those KS's receiving the message. This data field tagging may be requested by either a precondition evaluator which is about to instantiate a KS based on the values of particular fields (which represent the context satisfying the precondition), or by a KS process, once instantiated, which may request additional fields to be tagged.

For example, in its search through the global data base for conditions satisying the preconditions of some KS, a precondition evaluator may accumulate references to the data fields which it has examined; and when the entire precondition has been found to be satisfied, the precondition evaluator tags those fields (for which it has accumulated references) with the name of the KS process it is about to instantiate. Similarly, having been invoked, a KS process might wish to do certain computations, but only if certain data fields are not altered; the KS process itself can tag these fields and thereby be informed of subsequent tampering with the tagged fields. Subsets of these tagged fields (the subsets being formed according to the tagid's chosen) form *critical sets* (specifying the fields of the local context for the KS process) which are *locked* (see below) every time the KS process wishes to modify the global data base. Thus, after having locked the critical set and prior to performing any update operations, the KS process can check to see whether any other KS process has made any changes which might invalidate the current KS's assumed context (and hence, perhaps, invalidate its proposed update).[11] For example, if a KS process is verifying a hypothesis in the data base because its rating exceeds 50 (i.e., the rating value represents part of the local context for the KS process), then before performing any modifications on the data base which depend on the assumption, the KS process should check to be sure that no other KS process has invalidated the rating assumption in the meantime.

In addition to maintaining local copies of the various relevant changes that have occurred since the instantiation of a KS process, a KS process may also include in its context various *supersets* of these changes. For example, if a knowledge source is interested in the values of changes in a node's begin-time or end-time, it might be worthwhile to include a superset which accumulates indicators pointing to changes in either begin-time or end-time (since checking such a superset for new elements would be quicker than checking each constituent subset individually). Note that supersets do not themselves contain any data values, but rather are pointers to changes contained in some one of its subsets. Further note that a KS need not contain in its local context all (or any) of the constituent subsets of a superset in order to have that superset in its local context.

Data Base Deadlock Prevention: Any KS process may request exclusive access to some collection of fields at any time. Thus, unless some care is taken in ordering the requests for such exclusive access, the possibility of getting into a deadlock situation exists (where, for example, one KS process is waiting for exclusive access to a field already held exclusively by a second KS process, and vice versa, resulting in neither process being able to proceed). Applying a linear ordering to the set of lockable fields and requesting exclusive access according to that ordering is a commonly used means of deadlock avoidance in resource allocation. However, this technique works only if all the resources (fields) to be locked are known ahead of time. The ability to tag a data field allows a KS process to locate and examine in arbitrary order the set of hypotheses that will form the context for a data base modification and then, only after the entire set of desired hypotheses (and links) has been determined, to lock the desired set and perform the modification.

Regarding the global data base as having orthogonal dimensions of information level versus utterance time further allows a convenient means of locking entire regions of the data base without explicitly naming the nodes or fields to be affected. This "blanket-locking" is called *region locking*. For example, a KS process might request the locking of a region consisting of the time interval from 50 to 80 centiseconds at the phonetic level. To assist in de-

[9] The information which defines the change consists of the locus of the change (i.e., a node name and a field name) and the old value of the field.

[10] An alternative to replicating the change information for the various KS processes is to maintain a single central copy of those data, passing only references to the centralized items to the various local contexts. The individual change items may then be deleted from their central change sets whenever there are no further outstanding references to that change.

[11] The characterization of local contexts according to specific data fields (which is made possible in part by the choice of levels in the global data base) helps to minimize the overhead associated with context maintenance. Also, the smaller the context, the more flexible the scheduling strategy may be (since it needs to be less concerned with the time required for a context swap on a processor).

termining such regions, each node in the global data base contains a bit vector whose high-order bits describe the time regions covered by the node and whose low-order bits contain the integer level number of that node. These region fields may be manipulated to allow locking of various regions in the data base. Note that a region-lock may be used to obtain exclusive access to a time region which need not contain any nodes yet (i.e., one may region-lock a "vacuum").

To eliminate the possibilities of deadlock, the actual locking operation is relegated to a system primitive, and a KS process is required to present to the locking primitive the entire set of fields to which it wants exclusive access. This set is then extended to include all fields in the critical set of the calling KS process, for the reasons relating to context revalidation given above. The system locking primitive can then request exclusive access for this union of data fields, on behalf of the KS process, according to a universal ordering scheme (such an ordering being possible since every node in the global data base essentially has a unique serial number) which assures that no deadlocks occur. Having been granted exclusive access to all desired fields at once, the KS process may then check to see whether there have been any changes to the tagged data fields. If there have been none, it can proceed to perform its modifications (which modifications are sent to the local contexts of others who care about such things). However, if there were changes, the KS process can, after examining the changed data fields, decide whether it still wants to perform the modifications. When the KS process is finished updating the data base, it releases all its locked fields by executing a system primitive unlocking operator. In particular, the system ensures that a KS process will not request two lock operations without issuing an intervening unlock operation. In this manner, any possibility of a data base deadlock is eliminated.

Goal-Directed Scheduling: The computational sequence of a typical HSII run may be characterized by considering the chronological sequence of states of the utterance recognition at any particular data base level. For example, if one considers the efforts in producing the word level and traces the development of the "best" partial sentences, the processing that will have been done is analogous to a general tree-search, where each node of the tree represents some partially completed sentence (with the eventual resultant sentence being one of the leaves of the tree). The problem now is to guide this tree searching so as to find the answer leaf in an optimal way (according to some measure of optimality). To achieve this end, *ratings* are associated with every hypothesis and link in the global data base (and thus with every partial sentence node of the analogous search tree). Using these ratings (which are effectively evaluation functions indicating the goodness of the work done so far with respect to a given partial sentence), one may employ tree-searching strategies to advance the search in some optimal manner.

To complicate matters further, however, HSII is intended to be a multiprocess-oriented system. Therefore, schemes must be devised for effectively searching a problem-solving graph using parallel processes, since one can conceive of pursuing several branches of the search graph in parallel by asynchronously instantiating various KS processes to evaluate various alternative paths. One must also take into consideration the underlying hardware architecture. The physical placement of the global data base and the KS processes will have a very definite influence on the scheduling philosophy chosen and the resultant system efficiency.

To aid in making scheduling decisions, one may associate with every node in the global data base some *attention-focusing factors*, which are indicators telling how much effort has been devoted to processing in this area of the search tree and how desirable it is to devote further effort to this section of the tree. Such attention-focusing factors may also be associated with various speech time regions to indicate interest in doing further processing on certain regions of the utterance, regardless of any particular information level. Furthermore, attention-focusing factors are associated with every data base modification, thereby distributing attention-focusing factors to the various change sets which constitute the local contexts of the processes in the system. The scheduler is one such process which might be especially interested in such focusing factors, as will be described below. The use of the various ratings and attention-focusing factors allows HSII to perform *goal-directed scheduling*, which is process scheduling so as to achieve "optimal" recognition efficiency. The thrust of goal-directed scheduling is that, while there may be many processes ready to run and work on various parts of the search tree, one should first schedule those processes which can best help to achieve the goal of utterance recognition. Notice that such search-tree pruning techniques as the alpha–beta procedure (which is essentially a sequential algorithm anyway) do not apply to HSII's nongame search trees, which do not have the constraint that alternating levels of the tree represent the moves of an opponent.

Goal-directed scheduling may be viewed as having two separate functions: 1) using the ratings and attention-focusing factors associated with the global data base components to schedule *KS processes* which have been invoked (readied for execution) in response to events previously detected in the global data base, and 2) using these same attention-focusing factors to detect important areas in the global data base which require further work, and invoking *precondition evaluators* as soon as possible to instantiate new KS processes to work in those important areas. Thus, the attention-focusing factors within the global data base serve to schedule both KS processes and precondition evaluators.

A KS process might be scheduled for execution because it possesses the only processing capability available to be

applied to an important unexplored area of the data base. However, if there are many such processes ready to execute, the scheduler can perform a type of *means-ends* analysis in which those KS processes are scheduled which are likely to produce data base changes in which the system is currently interested (such interest being noted by high attention-focusing factors in a given change set). For example, if the data base contains focusing factors which highlight activity in a time region in which there are no structural connections between two adjoining levels, the scheduler would probably give a higher priority to a KS process which will attempt (as indicated in its external specifications) to make such a connection than to a KS process which is likely merely to perform a minor refinement on the ratings in one of the levels.

Another means of effecting goal-directed scheduling relates to the attention-focusing factors associated with various time regions of the utterance (such focusing factors reaching across all the information levels of the global data base). Using these time-region focusing factors, one can schedule KS processes which can contribute in a particular time region, or invoke precondition evaluators to instantiate some new KS processes to work within the desired time region.

SUMMARY, CURRENT STATUS, AND PLANS

In summary, HSII is a system organization for speech understanding that permits the representation of speech knowledge in terms of a large number of diverse KS's which cooperate via a generalized (in terms of both data and control) form of the hypothesize-and-test paradigm. KS's are independent and separable; they are activated in a data-directed manner and execute asynchronously, communicating information among themselves through a global data base. This global data base, which is a representation of the partial analysis of the utterance, is a three-dimensional data structure (in which the dimensions are level of representation, time, and alternatives) augmented by AND/OR structural relationships which interconnect elements of the data structure. This global data base structure permits information generated by one KS to be: 1) retained for use by other KS's, and 2) quickly propagated to other relevant parts of the data base. In addition to being a new representational framework for specifying speech knowledge, HSII is a system organization suitable for efficient implementation on a multiprocessor computer system. In particular, the system organization employs techniques which, while not violating the independence and modularity of KS's, permit: 1) avoidance of deadlock in the data base, 2) efficient implementation of data-directed sequencing of knowledge sources, and 3) goal-directed scheduling of asynchronously executable KS processes.

A preliminary, synchronous version of HSII has been operating on Carnegie-Mellon University's PDP-10 since January 1974. The fully asynchronous, multiprocess version of HSII is now in the final stages of being implemented, also on the PDP-10, and is expected to be running by August 1974. This multiprocess version of HSII will also contain the capability of simulating the effects of operating HSII in a multiprocessor environment. Experience with this multiprocess version of HSII, together with simulation data on the effects of operating in a multiprocessor environment, will form the basis for a multiprocessor version of HSII on a multi-miniprocessor computer (C.mmp). An initial implementation of HSII on C.mmp is expected to be completed in the first quarter of 1975.

A Carnegie-Mellon University technical report providing detailed examples of HSII's recognition process is planned for Autumn 1974.

APPENDIX I

HSI PERFORMANCE SUMMARY

Fig. 7 summarizes the performance of the HSI system as of Autumn 1973. The results are based on tests of 144 sentences containing 676 words, spoken by five speakers and using four different tasks (Chess, News Retrieval, Medical Questionnaire, and Desk Calculator) whose vocabularies range from 28 to 76 words. (More complete descriptions of the tasks are given in [5] and [12].)

Column 1 gives the data set number, task, and speaker identification. All of the speakers are male adults. Data set 4 was recorded over a long-distance telephone connection; the others were recorded in a fairly quiet environment using one of several medium to high-quality microphones. All of the utterances represent speech read from a script (as opposed to being spontaneous).

Column 2 gives the number of words in the task lexicon.

Column 3 shows the number of sentences in the data set; column 4 gives the total number of word tokens in the set.

Column 5 contains the results of recognition with just the Acoustics module (the acoustic-phonetic source of knowledge). The task lexicon is the sole restriction—the Syntax and Semantics modules are not used. The first subcolumn indicates the percent of times the correct word appears as the first choice of the Acoustics module in the hypothesis list. The second subcolumn indicates the percent of times the correct word is in the top two choices; the third shows the top three. In this mode each incorrect word recognition is overridden by specifying the correct word boundary positions (using carefully hand-segmented data) so that errors do not propagate.

Column 6 gives the results for the HSI system recognition with the Acoustics module and the Syntax module both operating. The first subcolumn indicates the percent of sentences recognized completely correctly. The "near-miss" (indicated below that number in the first subcolumn) indicates the percent of times the recognized utterance differed from the actual utterance by at most one word of

1	2	3	4	5 ACOUSTIC MODE			6 ACOUSTIC AND SYNTAX MODE				7 ACOUSTIC, SYNTAX, AND SEMANTIC MODE			
DATA SET: TASK/ SPEAKER	WORDS IN LEX	#SEN	#WORD	%IN #1	%IN #1 +#2	%IN #1 +#2 +#3	%SEN %NEAR MISS	%WOR	AVE T/SEN (Sec)	AVE T/S INP	%SEN %NEAR MISS	%WOR	AVE T/SEN	AVE T/S INP
1 CHESS/RN	31	14	62	61	79	85	43 100	87	11	6	100 100	100	9	5
2 CHESS/JB	31	19	86	44	69	78	74 95	93	12	9	100 100	100	8	6
3 CHESS/JB	31	21	105	39	68	73	15 50	69	15	8	48 90	88	13	7
4 CHESS/(TEL) BL	31	25	99	43	64	76	52 84	78	7	6	88 88	88	7	6
SUB-TOTALS		79	352	45	69	77	46 80	81	11	7	79 93	93	9	6
5 NEWS-RETREV/VL	28	19	104	38	53	67	21 42	63	13	7	XXX XXX	XXX	XXX	XXX
6 MED-QUES/JR	76	21	92	28	38	49	24 33	48	13	11	XXX XXX	XXX	XXX	XXX
7 DESK-CALC/JB	38	25	128	42	63	72	35 55	68	28	14	XXX XXX	XXX	XXX	XXX
TOTALS		144	676	39	61	71	38 64	78	15	9	79 93	93	9	6

Fig. 7. Performance results of the HSI system—Autumn 1973.

approximately similar phonetic structure. The second subcolumn gives the percent of words recognized correctly. The mean computation times on the PDP-10 computer (in seconds per sentence and in seconds per second of speech) are shown in subcolumns three and four.

Column 7 shows results for recognition using all three sources of knowledge (for the Chess task only): the Acoustics, Syntax, and Semantics modules. The subcolumns are similar to those of Column 6.

As one might expect, accuracy increases with the introduction of more KS's. It is also true that computation time decreases with more KS's; i.e., the additional KS's serve to reduce the size of the space that must be searched.

Performance differences across the different data sets are attributable to the following several sources:

1) Task lexicon size.

2) Phonetic content of the lexicons: many rules in the Acoustics module handle particular kinds of phonetic juxtaposition. Vocabularies as small as the ones used here do not exhaust all of these cases; each such vocabulary exhibits some cases not found in each of the others. Because a larger amount of effort has gone into the rules based on the Chess lexicon, that task also performs best.

3) Speaker differences.

It is encouraging that the performance using the telephone input (data set 4) does not appear to differ significantly from the other data sets, which use higher quality input. The labeling of acoustic segments is defined by a table of values derived from training syllables uttered by the speaker over the same channel used to input the test data; thus it is somewhat self-normalizing. Also, the system detects that there is very little higher frequency energy (in the case of the telephone input); this automatically triggers programmed readjustment of some thresholds and heuristics dealing with high frequency information.

ACKNOWLEDGMENT

We wish to acknowledge the contributions of the following people: A. Newell for his insights into the task of problem-solving, D. Parnas and L. Cooprider for their discussions on system design and decomposition, G. Gill for his untiring efforts in system implementation, and D. McCracken for his valuable criticism of the ideas as presented in this paper.

REFERENCES

[1] J. Baker, "The DRAGON system—an overview," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 22–26; also, this issue, pp. 24–29.

[2] J. Barnett, "A vocal data management system," *IEEE Trans. Audio Electroacoust.* (*Special Issue on 1972 Conference on Speech Communication and Processing*), vol. AU-21, pp. 185–188, June 1973.

[3] C. G. Bell *et al.*, "C.mmp: the CMU multi-miniprocessor computer," Dep. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, Pa., Tech. Rep., 1971.

[4] C. G. Bell, R. C. Chen, S. H. Fuller, J. Grason, S. Rege, and D. P. Siewiorek, "The architecture and application of computer modules: a set of components for digital systems design," presented at COMPCON 73, San Francisco, Calif., pp. 177–180, 1973.

[5] L. D. Erman, "An environment and system for machine understanding of connected speech," Ph.D. dissertation, Dep. Comput. Sci., Stanford Univ., Stanford, Calif., 1974.

[6] L. D. Erman, R. D. Fennell, V. R. Lesser, and D. R. Reddy, "System organizations for speech understanding: implications of network and multiprocessor computer architectures for AI," in *Proc. 3rd Int. Joint Conf. Artificial Intelligence*, Stanford, Calif., Aug. 1973, pp. 194–199.

[7] H. G. Goldberg, D. R. Reddy, and R. L. Suslick, "Parameter-independent machine segmentation and labeling," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 106–111.

[8] C. Hewitt, "Description and theoretical analyis (using schemata) of PLANNER: a language for proving theorems and manipulating models in a robot," Massachusetts Inst. Technol., Cambridge, project MAC, AI Memo. 251, 1972.

[9] F. Jelinek, L. R. Bahl, and R. L. Mercer, "Design of a linguistic statistical decoder for the recognition of continuous speech," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 255–260.

[10] M. J. Knudsen, "Real-time linear-predictive coding of speech on the SPS-41 microprogrammed triple-processor system," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 274–277; also this issue, pp. 140–145.

[11] S. Kriz, "A 16-bit A-D-A conversion system for high-fidelity audio research," in *Proc. IEEE Symp. Speech Recognition* Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 278–282; also, this issue, pp. 146–149.

[12] R. B. Neely, "On the use of syntax and semantics in a speech understanding system," Ph.D. dissertation, Stanford Univ., Stanford, Calif., 1973.

[13] A. Newell *et al.*, *Speech Understanding Systems: Final Report of a Study Group*, 1971. Amsterdam, The Netherlands: North-Holland, 1973.

[14] A. Newell, "Production systems: models of control structures," in *Visual Information Processing*, W. C. Chase, Ed. New York: Academic, 1973, pp. 463–526.

[15] D. R. Reddy, L. D. Erman, and R. B. Neely, "The C-MU speech recognition project," in *Proc. IEEE Conf. System Sciences and Cybernetics*, Pittsburgh, Pa., 1970.

[16] ——, "A model and a system for machine recognition of speech," *IEEE Trans. Audio Electroacoust.* (*Special Issue on 1972 Conference on Speech Communication and Processing*), vol. AU-21, pp. 229–238, June 1973.

[17] D. R. Reddy, L. D. Erman, R. D. Fennell, and R. B. Neely, "The HEARSAY speech understanding system: an example of

the recognition processes," in *Proc. 3rd Int. Joint Conf. Artificial Intelligence*, Stanford, Calif., Aug. 1973, pp. 185–183.
[18] R. Reddy and A. Newell, "Knowledge and its representation in a speech understanding system," in *Knowledge and Cognition*, L. W. Gregg, Ed. Washington, D. C.: Erlbaum, ch. 10, to be published.
[19] E. Rich, "Inference and use of simple predictive grammars," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, p. 242.
[20] H. B. Ritea, "A voice-controlled data management system," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 28–31.
[21] P. Rovner, B. Nash-Webber, and W. Woods, "Control concepts in a speech understanding system," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 267–272; also this issue, pp. 136–140.
[22] J. F. Rulifson *et al.*, "QA4: a procedural calculus for intuitive reasoning," AI Center, Stanford Res. Inst., Menlo Park, Calif., Tech. Note 73, 1973.
[23] L. Shockey and L. D. Erman, "Sub-lexical levels in the Hearsay II speech understanding system," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 208–210.
[24] W. A. Woods, "Motivation and overview of BBN SPEECHLIS: an experimental prototype for speech understanding research," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, p. 1–10; also this issue, pp. 2–10.