A RETROSPECTIVE VIEW OF THE HEARSAY-II ARCHITECTURE

Victor R. Lesser[1] and Lee *D. Erman*

Department of Computer Science[2]
Carnegie-Mellon University, Pittsburgh, Pa, 15213

ABSTRACT

The Hearsay model has heen presented as a paradigm for attacking errorful knowledge-intensive problems requiring multiple, cooperating knowledge sources. The Hearsay-II architecture is the latest attempt to explore the model. This paper describes experiences gained while successfully applying this architecture to the problem of speech understanding. The major conclusions are:

1. The paradigm of viewing problem solving in terms of hypothesize-and-test actions distributed among distinct representations of the problem has been shown to be computationally feasible.
2. A global working memory (the "blackboard"), in which the distinct representations are integrated in a uniform manner, has made it convenient to construct and integrate the individual sources of knowledge needed for the problem solution.
3. The use of a uniform data-directed structure for controlling knowledge-source activity has made the system easy to understand and modify.
4. A solution has been demonstrated to the problem of focus-of-attention in this type of control environment. This solution does not need to be modified when the sources of knowledge in the system are changed.

INTRODUCTION

The Hearsay model [Red73Mo] has been developed for problem-solving in domains which must use large amounts of diverse, errorful, and incomplete knowledge in order to search in a large space. The *Hearsay-1* architecture and system [Red73Hx and Erm74En] represented a first (and successful) attempt to apply that model to the problem of understanding connected speech in specialized task domains. In this first application, the size of the vocabulary (less than 100 words) and complexity of the grammar were very limited.

Experiences with Hearsay-1 led to the more generalized *Hearsay-II* architecture [Les750r and Erm75Mu] in order to handle more difficult problems (e.g., larger vocabularies and less-constrained grammars). The first configuration of knowledge sources (KSs) for Hearsay-II -- configuration CI — was complete in January, 1976 [CMU76W4]. This implementation had poor performance (e.g., 107 sentences correct in 85 MIPSS (million instructions per second of speech) on a 250-word vocabulary). Experience with this configuration has led to a substantially different set of KSs — configuration C2 [CMU77Su]. This configuration performs substantially better (e.g., 857. correct in 60 MIPSS on a 1,000-word vocabulary).

The Hearsay-II system, with the second configuration, has been successful: it comes close to the original performance goals set out in 1971 to be met by the end of 1976 for the ARPA speech understanding effort [New73Sp] and does so with a system organization that is of interest because of the potential for its application to other problem areas. Several other problems have been attacked with organizations strongly influenced by the Hearsay-II structure: image understanding [Pra77Se], reading comprehension [Rum76To], protein-crystallographic analysis [Eng77Kn], signal understanding [Nii77Ru], and complex learning [Sol77Kn].

This paper is divided into two major parts. The first part presents an overview of the Hearsay model, the Hearsay-II architecture, which is a further specification of this model, and the two KS configurations. (More detailed descriptions of these configurations are contained in the appendix.) The second part of the paper discusses the implication of these experiences for the Hearsay model and the Hearsay-II architecture. In particular, those aspects of the architecture are identified that have contributed most strongly to the success of the system, as well as those parts that need the most future work.** This discussion is structured around two themes -- the multi-level global data base (blackboard) for KS cooperation, and the asynchronous, data-directed control structure for KS activation.[5]

OVERVIEW OF THE HEARSAY MODEL

A number of characteristics of the problem drive the Hearsay model:

1. Large search space.
2. Diverse sources of knowledge. Many of the KSs are large; some have large internal search problems of their own.
3. Error and variability. These are characteristics of both the input data (the acoustic signal) and the processing of knowledge sources.
4. Experimental approach needed for system development. This implies the need for iterating the system and running over large amounts of data.
5. Performance requirement -- accuracy and speed. This is true of any practical solution to the problem as well as during development (because of the experimental nature).

The basic notions of the Hearsay model [Red73Mo] were developed in response to the requirements just stated:

1. The KSs are kept separate, independent, and anonymous. This separation is felt to be a decomposition which is natural and also can help make the combinatoric problems more tractable. For development purposes, the separation should help with system modifications (especially adding and modifying KSs) and evaluation.
2. A global data structure -- the *blackboard* -- is the means of communication and interaction of KSs. This provides an hypothesize-and-test means of interaction. Each KS accesses and modifies the blackboard in a uniform way.
3. A KS responds to changes to the blackboard which it is concerned with; it applies its knowledge within the context'

1 Author's current address: Department of Computer and Information Science, University of Mass., Amherst, Mass. 01003.

2 This work was supported by the Defense Advanced Research Projects Agency (F44620-73-C-0074) and is monitored by the Air Force Office of Scientific Research.

3 Other approaches for solving this class of problem include production systems, frames [Min74Fr], heterarchical structures [Wal770v and Woo76Fi], relaxation techniques [Bar76MS and Ros76Sc), Planner [Hew72De], QA4 [Rul73Qa], and the Locus model [Low76Ha and Rub77Lo].

4 The fact that certain parts of the implementation need further work does not necessarily indicate deficiencies with the basic Hearsay model, but rather points out inadequacies in the Hearsay-II implementation of the model. It is to the model's credit that even though some of its more sophisticated capabilities are not implemented effectively, it still provides an appropriate framework for the successful solution of a complex task. Thus, one of the intents of this paper is to define some of the major design goals for the next iteration in the implementation of the Hearsay model.

5 While this paper discusses the means of organizing the knowledge and applying it to the problem, it does not describe in detail nor quantify the knowledge in the system. At least as much work has been expended on specifying and debugging the knowledge in the system as on building and refining the structure to hold and apply that knowledge.

of such a change. This implies data-directed activation of KSs.

OVERVIEW OF THE HEARSAY-II ARCHITECTURE

The Hearsay-II architecture is one framework for implementing the Hearsay model. In this section, a very brief overview of that architecture is given. More details are described in [Les750r and Erm75Mu].

The Blackboard

The blackboard is partitioned into distinct information *Levels;* each level is used to hold a different representation of the problem space. (Examples of levels are "phrase", "word", "syllable", and "segment".) The decomposition of the problem space into levels is a natural parallel to the decomposition of the knowledge into separate KSs. For most KSs, the KS needs to deal with only a few (usually two) levels to apply its knowledge. Its interface to the rest of the system is in units and concepts that are natural to it.

The sequence of levels forms a loose hierarchical structure in which the elements at each level can be described approximately as abstractions of elements at the next lower level. The possible hypotheses at a level form a problem space for KSs operating at that level. A partial solution (i.e., a group of hypotheses) at one level can be used to constrain the search at an adjacent level. For example, consider a KS which can predict and rate words based on acoustic information and another KS which knows about the grammar of the language. The first KS can generate a set of candidate word hypotheses. The second KS can use these hypotheses to generate phrase hypotheses which can be used, in turn, to predict words likely to precede or follow. These predictions can now constrain the search for the first KS.

Associated with each level is a set of primitive elements appropriate for representing the problem at that level; e.g., the elements at the word level are the words of the vocabulary to be recognized. The major units on the blackboard are *hypotheses.* An hypothesis is an interpretation of a portion of the spoken utterance at a particular level. E.g., an hypothesis might represent the assertion that the word "GIVE" was spoken at the beginning of the utterance. Each hypothesis at a given level is labeled as being a particular element of the set of primitive elements at that level.

Each hypothesis, no matter what its level, has a uniform attribute-value structure. Some attributes (and values) are required of all hypotheses and others are optional, as needed. Included among the required attributes of an hypothesis are its level (e.g., word), its element name (e.g., "GIVE"), and an estimate of its time coordinates within the spoken utterance (which can include notions of "fuzziness" of estimate). The level and time attributes place a two-dimensional structure on hypotheses which partitions the blackboard and can be used for addressing hypotheses. Note that two or more hypotheses at the same level with significantly overlapping times are *competitors]* i.e., they represent competing interpretations of a portion of the utterance.

Other attributes of an hypothesis include information about its *structural relationships* with other hypotheses (forming an AND/OR graph), *validity ratings* (i.e., estimates by KSs of the "truth" of the hypothesis), and *processing state.* The processing state attributes are summaries and classifications of the other attributes. E.g., the values of the rating attributes are summarized by the "rating state" attribute that takes a value from the set "Unrated", "Neutral", "Verified", "Guaranteed", or "Rejected". New attributes can be created by any KS and may be used for passing arbitrary information about an hypothesis between instantiations of the same or different KSs.

A KS can create new hypotheses, specifying values for attributes of the new hypothesis. Given the "name" of an hypothesis, a KS can examine or modify attributes of that hypothesis. In addition, sets of hypotheses may be retrieved associatively, based on the values of their attributes (e.g., all hypotheses at the syllable level whose durations are greater than 250 msec). The hypothesis structure is uniform across all levels in the blackboard. Thus, the form of access and modification to hypotheses by KSs can also be uniform and is accomplished by calling Kernel procedures; the set of these procedures comprises the *blackboard handler.*

In addition to the information in each hypothesis which can be accessed by KSs, *auxiliary state information* is maintained by the blackboard handler in specialized data structures. Examples of this information are (1) a representation of hypotheses at each level arranged for efficient associative retrieval by time and (2) the name of the highest-rated hypothesis in each time area. These auxiliary structures are updated by the blackboard handler automatically as KSs make changes to the blackboard.

Structure of Knowledge-Sources

Each KS has two major components: a *precondition* and *an action.* The purpose of the precondition is to find a subset of hypotheses that are appropriate for action by the KS and to invoke the KS on that subset; the subset is called the *stimulus frame* of the KS instantiation. For example, the precondition of the KS that generates word hypotheses based on syllables looks for new syllable hypotheses. When invoking the KS, the precondition provides the system scheduler with, in addition to the stimulus frame, a stylized description of the likely action that the KS instantiation will perform (if and when it is allowed to execute); this estimate of action is called the *response frame.* For example, a response frame for the syllable-based word hypothesizer (MOW) indicates that the action will be to generate hypotheses at the word level and in a time area that includes at least that of the stimulus frame. The action part of a KS is a program (written in SAIL [Rei76SA]) for applying the knowledge to the stimulus frame and making appropriate changes to the blackboard. In general, the changes made will serve to trigger more KS activations.

To keep from having to fire the precondition continuously to search the blackboard, each precondition declares to the blackboard handler in a non-procedural way the primitive kinds of blackboard changes in which it is interested. Each precondition is triggered only when such primitive changes occur (and is then given pointers to all of them). This changes a polling action into an interrupt-driven one and is more efficient, especially as the number of preconditions gets large. After being triggered (and when scheduled for execution), the precondition (also a SAIL procedure) can do arbitrary searching of the blackboard for hypothesis configurations of interest to its KS.

Several KSs may be grouped together into *modules.* The KSs within a module may share code and long-term built-in data. A discussion of the module construct, including its implications for KS independence, is given below in the section on "KS Independence".

Scheduling

Whenever a precondition is executed, it checks all blackboard events in which it is interested that have occurred since the last time it executed. For example, a "new hypothesis" to a precondition is any hypothesis which was created between the last time the precondition executed and its current execution. Thus, a precondition may be thought of as executing, then "sleeping" for a time while retaining state, then waking (executing again) and being able to find all new events of interest to it.

However, whenever a *KS* executes, it uses the stimulus frame specific to that invocation. Each KS execution goes to completion; that is, the KS cannot put itself to "sleep", waiting for some other event (on the blackboard) to occur.

At any point, there are, in general, a number of pending tasks to execute — both invoked KSs and triggered preconditions. (In practice, the number of pending tasks often exceeds 200.) A *scheduler* in the kernel [Hay77Fo] calculates a priority for each waiting task and selects for execution the task with the highest priority. The priority calculation attempts to estimate the usefulness of the action in fulfilling the overall system goal of recognizing the utterance. This estimation is based on the specific stimulus and response frames of the actions and on overall blackboard state information, which includes such notions as the best hypotheses in each time area in the utterance, and how much time has elapsed since the current best hypothesis was generated.

The priority of a KS is recalculated if the validity of its stimulus frame is changed or the auxiliary state pertinent to evaluating the significance of the response frame is modified.

Some KSs are not directly involved in hypothesizing and testing partial solutions; instead, these control the search by influencing the activation of other KSs. These *policy* KSs can be used to impose global search strategies on the basic priority scheduling mechanism.

THE CONFIGURATIONS

Following are brief overviews of configurations CI and C2, to provide a basis for subsequent discussion. The appendix contains more detailed descriptions of the KSs, as well as pointers to published papers.

Figure 1 gives a schematic of configuration C1 as it was operational in January, 1976. The levels are indicated by solid horizontal lines and are labeled at the left. KSs are indicated by vertical arcs with the circled end indicating the level where its stimulus frame is and the pointed end indicating the level of its response frame. The name of a KS is connected to its arc by a dashed horizontal line. As segment hypotheses were generated from the acoustic data (SEG), they might be combined to form larger segment hypotheses (CSEG). Phone hypotheses were created, based on one or more contiguous segments (PSYN). Syllables were predicted from the phones (POM) and words from the syllables (MOW). Phrase hypotheses were constructed from contiguous word or phrase hypotheses which were syntactically consistent (RECOG). Other KSs (PREDICT, RESPELL, and POSTDICT) accomplished various syntactic extension and prediction functions at the phrase and word levels. Verification of predicted words was carried out by expanding the words into their expected syllables (WOM), expanding the syllables into expected phonemes (MOS), and matching the sequences of expected phonemes with the recognized phones (TIME and SEARCH). Changes of ratings of hypotheses were propagated to structurally connected hypotheses (RPOL). The FOCUS policy KS controlled the search by setting priorities for various kinds of KS actions.

Figure 2 gives a schematic of configuration C2 as it was operational in September, 1976. First, all segment hypotheses are generated from the parametric representation of the acoustic signal (SEG). Next, syllables are predicted from the segments (POM). Then, words are predicted from the syllables (MOW); the most likely words in each time interval placed on the blackboard (WORDCTL). Next, a heuristic word-sequence hypothesizer (WOSEQ) attempts to identify the most probable sequences of word hypotheses (consisting of successive language-adjacent word pairs). Because this KS exploits statistical methods to improve credibility, the initial word sequence hypotheses are much more accurate than are hypotheses based on single words. Subsequently, KSs are invoked to attempt to parse the hypothesized word sequences to determine if they are grammatical (PARSE), to predict possible time-adjacent grammatical word extensions (EXTEND), to hypothesize and verify new words satisfying these predictions (MEW), to concatenate grammatical and time-adj-acent word sequences (CONCAT), to propagate ratings (RPOL), to reject phrases and to determine when the serach should be terminated (STOP), and to generate new word sequence hypotheses (WOSCTL).

The major system-related differences between these configurations[6] are listed here; they will be discussed individually throughout the paper.

1. CI has asynchronous processing throughout. C2 has an initial pass of sequential, bottom-up processing to the word level; i.e., all segments are created, then all syllables, then a selection of words.
2. CI used the blackboard extensively for intra-KS state-saving between instantiations of a KS (e.g., SEARCH and RECOG-PREDICT-RESPELL-POSTDICT). in C2, this was greatly reduced, with KSs doing more computation internally and in larger units (e.g., MEW and PARSE-EXTEND-CONCAT).
3. C2 generated simpler hypothesis networks than those in CI. For example, SEARCH and TIME built complex structures to represent verifications of words; MEW builds very simple ones for the same purpose.

EXPERIENCES WITH HEARSAY-II

This section addresses the following questions: How well did the Hearsay-II system meet its original design goals and were these goals appropriate for problem solving in the speech understanding domain (and more generally in errorful domains which require extensive search)? This discussion is based on approximately three years of experience with the Hearsay-II architecture, including numerous iterations of both the system architecture and KS configurations/ These questions will be discussed in the context of two major aspects of the Hearsay-II architecture: the blackboard global data base, and KS interaction and control.

Blackboard Data Base

There are two major design themes reflected in the structure of the blackboard. The first theme is the avoidance of expensive and complicated backtracking control structures by the representation of alternative, distributed hypotheses in an integrated multi-level manner. The second design theme is the representation of all information levels with a high-level, uniform structure, in order to allow all KSs to contribute their information to the blackboard in an identical and anonymous manner.

Distributed Representation

It was hoped that the first design theme would (a) avoid the redundant calculation of previously-generated results and (b) allow KSs to apply their knowledge selectively to places in the blackboard where further processing would resolve contradictory evidence supporting likely, alternative hypotheses.

The ability to save partial results on the blackboard in an integrated manner, in terms of hypothesis sub-networks, has been a very positive characteristic of the architecture; it avoids a significant amount of unnecessary recalculation of results previously generated. This was especially true for KSs operating at the word and phrase levels. This was also true for KSs in the CI configuration operating at lower information levels, for example, the TIME and SEARCH KSs. However, later versions of these KSs (e.g., MEW in C2), for reasons of efficiency (to be discussed later), do not save partial results on the blackboard.

The use of an integrated representation as a way of efficiently resolving competition among KSs wanting to work on the same hypotheses has not been exploited, nor has the ability to bring to bear specialized knowledge dynamically to resolve the conflict among competing, alternative hypotheses (for example, a specialized KS to resolve ambiguity between two word hypotheses

6 Though we are here concerned with systems issues, it is worth pointing out that WOSEQ is a novel KS which significantly contributes to the success of C2. It limits the search space by providing large hypotheses which act as islands of reliability and bases for further search. This KS uses approximate syntactic knowledge to examine efficiently many alternative sequences of low-reliability word hypotheses and generate a small number of more reliable phrase hypotheses.

7 The emphasis on the two configurations as fixed points can be misleading; rather than appearing full-grown, the configurations evolved over time, with numerous iterations required first to develop CI and then C2 from CI.

8 Hayes-Roth [Hay77Ro], in discussing how to evaluate the potential usefulness of a KS action, introduces the concept of *diagnosticty* as an important component in a KS priority function. Diagnosticity is a measure of how much contradictory evidence could potentially be resolved by a particular KS action.

9 The usual manner of accomplishing this is having each KS, as it is about to create a new hypothesis, first check that a hypothesis does not already exist which is sufficiently similar to the one it is about to create.

that are very close acoustically — e.g., "sit" and "split"). In addition, the ability given by the integrated representation to re-evaluate automatically (i.e., without KS intervention) an hypothesis* credibility when its supporting environment is modified is not exploited in the C2 configuration (although it was C1). In the C2 configuration, hypothesis credibility is never modified in an explicit sense; rather, new and different hypotheses are created. A side effect of this approach is that hypotheses are n*ever* deleted from the blackboard.

One explanation for the lack of full use of the integrated, multi-level representation of hypotheses could be just that the particular task domain of speech understanding does not need these capabilities. However, it is our feeling that there are fundamental weaknesses in the Hearsay-II representation of an integrated, multi-level hypothesis; these weaknesses (to be discussed below) make it difficult, both in terms of execution time and programming complexity, to perform the desired analyses of the hypothesis structure and its surrounding environment. This type of analysis is the key to the effective use of the sophisticated processing capabilities that are possible within the framework of the Hearsay model.

Hypothesis Network Structure

A major problem in using the blackboard is that one cannot operate on a network (in its simplest form, a tree) of interconnected hypotheses as a composite unit. There is a basic confusion in Hearsay-IPs implementation of hypothesis networks between (a) the hypothesis at the top of the tree (the highest level of interpretation) and (b) the whole tree; the state information associated with an hypothesis is very local and does not adequately characterize the state(s) of the hypothesis network(s) connected to it. In order to operate effectively in a distributed manner on interconnected multi-level hypothesis networks, the state information associated with an individual hypothesis must allow a KS to analyze quickly the local environment of an hypothesis and, more importantly, the role that the hypothesis plays in the larger context of the hypothesis networks it is part of. One of the consequences of this deficiency is the difficulty encountered in making appropriate scheduling decisions because the more global import of a potential KS action cannot be determined easily.

For example, in configuration C1, an hypothesis at the phrase level was constructed out of hypotheses at the phrase, word, syllable, surface-phoneme, phone, and segment levels. Because of the asynchronous nature of processing, a phrase hypothesis could be supported by word hypotheses in different stages of verification -- some might be fully verified, others only partially verified, or some totally unverified. Possible KS actions waiting to work on this hypothesis network could be a separate verification of each unverified word, an attempt to extend the phrase in either the right or left direction, a search for co-articulation effects among different word pairs, or a full verification of a partially verified word. These actions represent processing at different information levels. Given the existing hypothesis interconnection primitives, there is no way to determine easily that all these actions relate to the same hypothesis network, nor what import each action could potentially have in judging the credibility of the entire network.

Another symptom of this problem is the inability to express, except in a very limited way, what type of processing has already been applied to* an hypothesis network and what further processing could possibly be applied. This inability again impacts the scheduler because it makes it difficult to schedule "competing" KSs (i.e., KSs which could work on the same or different parts of a specific hypothesis network) appropriately. Because of these difficulties there has been, in later KS configurations, only a very limited (and simply represented and analyzable) form of KS competition.

Another aspect of the inadequate network structure is that the primitives for specifying structural relationships between hypotheses require many intermediate levels to represent certain types of connectivity patterns. This need for many intermediate levels is expensive in in storage space and, more importantly, time; it requires a great deal of network searching through the connection structure to analyze the relationship of an hypothesis to its immediate surrounding environment. These intermediate levels represent a level of detail which is unnecessary for some types of KS analysis and which interfere with these analyses by making them unwarrantedly complex. Once it has been constructed, it is impossible to bypass this level of detail in situations in which it is not pertinent. For example, an information level may contain many intermediate sublevels built out of the connection primitives; a KS using information at this level may want only to examine those hypotheses which are the highest sublevel in each time area. This type of operation, given the current blackboard retrieval primitives, requires the examination of all hypotheses in a specified time area. Another complication of not being able to hide these intermediate levels is that a KS in some cases has to know the exact structure of the intermediate levels used by another KS in order to be able to skip over them, thus making the KSs less independent.

In summary, the experience to date on the distributed representation approach indicates that the implementations of this concept explored so far are neither general nor efficient enough in two major interrelated aspects -- how hypotheses can be combined into a network and how the state information associated with an individual hypothesis reflects the hypothesis networks connected to it. To elaborate further, what is missing from the blackboard structure is a way of viewing the shared network structure from a different perspective. This perspective should permit the particular path through the network that defines a specific composite hypothesis to be both viewed in isolation from other paths that are intertwined with it, and also in a way that eliminates superfluous sub-structure. From this type of perspective, the importance of potential KS actions could be judged efficiently and related to the history of previous processing.[11]

Uniform Blackboard Structure

Let us now examine the second major design theme used to structure the blackboard: a uniform structure at all information levels. From a programming point of view, both in terms of KS writers and system implementors, the uniform structure of the blackboard has been a good design choice. By having a uniform structure, a variety of standard blackboard creation, accessing, display, analysis, and debugging functions could be developed that are usable by all KSs. These standard functions, some of which are quite complex, make it convenient for a KS writer to interface his knowledge source with the system. The ease with which this interfacing could be accomplished is exemplified by the fact that, in a period of six months, configuration C2, which is almost entirely new relative to C1, was developed and debugged. Because of this uniform structure of hypotheses and their connections, it is often possible for a KS to be recoded so that it generates a different local hypothesis structure without requiring the recoding of other KSs in the system; this is true because a KS can probe the blackboard with sophisticated built-in retrieval operations which, in many cases, shield the KS from changes made by other KSs. For example, there is the *structural-adjacency* blackboard primitive which, given a hypothesis, finds all hypotheses at a particular information level that are immediately adjacent to the given hypothesis based on the AND/OR connection structure among hypotheses.

The uniformity of the attribute structure of hypotheses also

10 It is expensive to trace through an hypothesis network to determine the global import of a potential KS action. But this cost is not unreasonable relative to the total system execution time for a configuration which contains KSs that perform moderately large amounts of internal computation. However, the major computational expense comes in dynamically updating the global import of a pending KS action as modifications are made to the blackboard since there are a large number of these modifications: it is necessary both to find which waiting KS instantiations have priorities that are affected by the modification and then to recalculate the priorities for those affected.

11 A possible approach for implementing this different type of perspective is discussed in work by Hendrix [Hen75Ex] on partitioned semantic networks.

makes it possible to monitor efficiently for blackboard changes which are to trigger preconditions. Each precondition needs only to declare to the blackboard handler the names of the attributes at each level in which it is interested. When an attribute is changed, the blackboard handler then triggers all preconditions interested in it.

The uniform blackboard structure, though efficiently implemented, is not appropriate as a scratchpad for the internal computations of a KS. This type of use of the blackboard is often inappropriate because its uniform, general structure does not come completely free in the storage requirements for an hypothesis and the cost of creation and access; most internal computations of a KS do not need this generality. An example of a misuse of the blackboard was the case of the syntax analyzer knowledge source, SASS [Hay77Un], In early versions of this KS, the blackboard was used to hold the partial parse trees developed in attempting to parse a language fragment; current versions of this KS, which use a tailored, internal data structure for parsing, are two orders of magnitude faster than the original blackboard-based version of this KS. This case history seems to confirm the notion that there are advantages to specialization of structures: one for KS interaction (i.e., the blackboard), and separate ones for each KS.

The blackboard has also proven to be useful as a data base for the scheduler [Hay77Fo]. Because of the uniform hypothesis structure, instantiations of KSs can specify scheduling information in a uniform way (as stimulus and response frames), allowing new KSs to be introduced without having to modify the scheduler. The representation of alternative hypotheses in an integrated, uniform fashion also makes it possible to compare directly the pending KS instantiations to determine which will likely contribute most to further progress; the scheduler 1) can determine those areas on the blackboard that most need further work and locate the pending KS instantiations that are relevent to those areas and 2) estimate the amount that a KS instantiation will improve the quality of hypotheses in the area of its action.

Long-Term Information Structures

Associated with each information level of the blackboard, there is, as previously discussed, a set of primitive elements that are used to label hypotheses at that level. The kernel interface provides facilities for creating, accessing, and displaying these labels. In addition, arbitrary data structures can be associated with each label. These structures, for example at the word information level, can be simple, such as the average expected duration of each word, or complex, such as a network which specifies alternative syllabic spellings for each word. In the complex case, this structure often is used to relate labels at one information level with labels at another; this relationship is used by a KS which operates between different levels (e.g., in the example given here, WOM in configuration CI). These data structures related to labels constitute much of the long-term (built-in) KS-defined information structures of the system and often represent most of the problem-specific knowledge in a KS.

Each KS (or group of KSs) defines whatever ad hoc structure seems appropriate for the particular kind of information to be represented. There has been no attempt to define a uniform set of kernel interfaces for creating and accessing these long-term data structures, nor a set of relationships (connection primitives) for relating labels at different levels. However, it seems possible to attempt to define a small number of representations within the kernel; these structures would mimic the hierarchical structure of the blackboard. (Hanson and Riseman in their work on image understanding have a system architecture [Pra77Se] very similar to the blackboard and have included a complementary long-term memory structure.)

The major drawback of not having a predefined long-term memory is that if KSs want to share this information they have to agree among themselves upon a specific structure, thus violating independence considerations. In addition, uniform structures could make KSs easier to understand, develop, and analyze.

On the other hand, these long-term structures must be highly optimized because of their large size and the high frequency with which they are accessed.[12] The approach taken of tailoring these structures to the particular KS(s) using them allowed for efficient implementations in terms of both time and space. It is also possible that explicit tailoring has led to KSs which are easier to understand than if they were forced to fit their requirements into a uniform structure.

Thus, there are still open questions about the desirability of providing uniform structures for representing the knowledge in KSs; hopefully, future implementations will explore these possibilities.

Conclusions About Blackboard Usage

In trying to draw some conclusions about our experiences with the use of the blackboard, the main issue that constantly comes up is time and space efficiency. In errorful task domains, such as speech understanding, a large number of alternative interpretations of the data must be examined and analyzed. The blackboard concept is effective in the Hearsay-II implementation to the degree that it allows this search to be efficient. Analysis of the CI configuration indicated that certain types of KS processing on the blackboard were not efficient. Reimplementation of the KSs in order to eliminate those types of processing resulted in the C2 configuration. The major uses of the blackboard in the C2 configuration are:

1. A storage, area for high-level intermediate results generated by the search. This storage area avoids the unnecessary recalculation of these results if they are encountered on future search paths.
2. A communication area for KSs, with strong and simplified assumptions by a KS of what structures can be generated by other KSs.
3. A data base for the scheduler.
4. A common display, debugging, and performance evaluation area.

Knowledge-Source Interaction and Control

The asynchronous, data-directed control structure used In Hearsay-II was designed to permit:

1. The quick refocusing of attention to appropriate hypotheses in the blackboard.
2. The flexible reconfiguration of the system with different sets of independent (and possibly competing) KSs, and different global control strategies.
3. The exploration of parallel processing.

This section will examine each of these requirements along two dimensions: Were the capabilities embodied in the requirement important to the project, and how well did the control structure (in terms of time, space, and ease of representation) implement these capabilities?

Appropriateness of a Data-Directed Control Structure

The first requirement, quick refocussing, was based on the following model for processing in the speech domain. Processing can be organized in terms of the incremental additions of small units of information to a limited number of alternative hypotheses. The limited number of alternatives derives from the view that there are islands of reliability in the acoustic data that can be used to anchor the search. Each small increment of information should help to verify, refute, or augment (expand) an hypothesis. A KS action, though performed in a local context, could also have the side effect of contributing information useful in the evaluation of alternative hypotheses (i.e., in other contexts). Thus, after each incremental addition of information (through the execution of a KS), it is necessary to re-examine the set of potential actions that now can be activated and determine which of these will most likely resolve

12 For example, the description of the grammar used by the KSs within the SASS module in configuration C2 is a network of 3100 nodes. Each node has about seven pointers to other nodes, plus several pieces of auxiliary information. A typical KS action, e.g., parsing a four-word phrase, might make 100 to 5,000 node accesses.

ambiguity. An asynchronous, data-directed architecture makes it convenient to implement such a processing strategy by permitting KS action to be directed by the data: it delays the application of Knowledge until there is enough information for a meaningful result (decision), and it re-applies the Knowledge when, at a later time, additional information is generated that bears on the original decision.

In those parts of the blacKboard where processing followed this model, the data-directed control structure was *very* effective. However, at lower levels of speech processing (i.e., segmentation and labeling, syllable hypothesis generation based on segments, and word spotting based on syllables), this model was found to be inappropriate because there is not enough reliability in credibility scores of hypotheses to form hypothesis islands that can reliably anchor the search. Thus, processing at these levels cannot be selective (depth-first), and instead requires a complete scan (breadth-first), for which asynchronous control has no advantages (and considerable costs).

A major change in going from configuration CI to C2 was maKing the lower levels of processing more sequential and bottom-up. Not until the word level is reached do hypothesis credibility scores have enough reliability to justify the more complex processing required of an asynchronous, data-directed control structure. The presence of these islands of reliability is in itself not a sufficient condition for the use of this sophisticated control structure. What is additionally required is that there is either a significant cost to evaluate each alternative or a large number of alternatives (combinatoric explosion in the search space); only then is the overhead involved in implementing a data-directed control structure worthwhile.

In addition to the control overhead, an asynchronous control structure requires a more complex internal structure for a KS. This complexity arises because, as new information is asynchronously generated, a KS must have the additional logic to determine whether this new information allows it to make a decision it could not previously maKe or whether this information contradicts a previous decision. In the latter case, it must modify the previous decision, which may involve modifying decisions made as a consequence of the original one. Where processing involves a complex hypothesis networK structure with much detailed structure, the nature of asynchronous processing in response to a change at the detailed level is costly, both in terms of processing time and complexity of the KS, and should be avoided unless the compensatory benefits are large. As previously mentioned, the inadequacies in the blacKboard structure which maKe it difficult to sKip over detailed structure exacerbate these problems. (The SEARCH KS in configuration CI is an example of a KS worKing asynchronously at a detailed level. Although the acoustic-phonetic Knowledge applied by SEARCH was represented by a relatively simple data structure within the KS, the code necessary for examining and incrementally building large, integrated, and competing AND/OR structures on the blacKboard was very complex and the number of KS executions needed to verify a word was large -- on the order of ten to one hundred. In C2, the function of word verification was replaced by the MEW KS -- here, verifying a word is an atomic act (as far as other KS actions are concerned) and is carried out using tailored structures internal to the KS. Each execution of MEW forced a recalculation of the detailed structure, rather than sharing such structures across executions.)

Overhead Costs of Data-Directed Control

The overhead cost of implementing an asynchronous data-directed control structure for computation of medium level granularity (i.e., a KS action which involves greater than 1/10 second of internal computation) is not significant. The major cost involves monitoring each modify operation to the blacKboard to determine whether any preconditions are interested in being notified of this specific change. This cost of monitoring and notification makes a modify operation 12 times as expensive as a read operation. However, in the C2 configuration there are 29 times as many reads as modify operations, thus maKing this aspect of implementing a data-directed control only 4% of the total cost of a run.

Another cost associated with implementing this type of control structure involves maintaining a scheduler queue of waiting KS instantiations and performing priority calculations to decide which instantiation to run next. However, these focus of control calculations, possibly expressed in a different way, are necessary in any problem-solving system that involves a dynamic search. The more general implementation of these calculations in the context of an asynchronous control structure does not appear to generate significantly more system overhead than a specialized implementation of them in a system with more explicit control structure. The cost of maintaining and updating the scheduler queues and calculating the priorities was about 57 to 77 of a total run.

Further costs involved in implementing this type of asynchronous control structure arise because of the delay between the invocation of a KS and its execution. The KS must, in general, contain code that revalidates its invocation context before beginning execution. However, by maKing some assumptions about the type of processing other KSs could effect at particular information levels, there was in practice very little need for context revalidation. KSs did not in general interact by modifying previously-made assumptions and detailed structures constructed by other KSs, but rather through the incremental addition of new hypotheses to existing structures or the verification of previously unverified hypotheses.

KS Independence

As indicated above, complete independence among KSs was not accomplished. However, information about the processing characteristics of other KSs is generally *very* restricted, and relates only to KSs which share either dynamic information on the blacKboard or long-term static information. To facilitate such data sharing, the concept of a *module* was introduced into the architecture. A module contains a set of preconditions and KSs which share common structures and related accessing procedures. The KSs contained in a module generally operate at the same or adjacent information levels and thus also share specialized accessing and display routines for these information levels of the blacKboard. A module usually represents the code of one KS programmer and typically contains one to four KSs and one to four preconditions. The clustering of KSs by their long-term information structures turned out to be a convenient decomposition for separably instantiable but related activity. The KS module is the atomic unit which is the basic building blocK for different KS configurations.[13]

How important is the property of independence of KSs? For the two configurations discussed here, the KS modules are not completely independent. However, during the lifetime of the project, which involved numerous iterations of KSs, there has been very little difficulty encountered by this lacK of complete independence (i.e., the "subroutine interaction problem" did not haunt us). It has been possible to configure systems with subsets of KS modules (e.g., a "top-end" system that deals only with word and phrase hypotheses or a "bottom-end" system which deals only at and below the word level) without modifications to the modules involved.

The reason for having little difficulty with the subroutine interaction problem can be traced to the data-directed activation of KSs. In general, interaction among KSs is accomplished by having a KS modify the attribute structure of an hypothesis in a way which causes some other KS(s) to be activated and attend to that hypothesis. In order for KSs to communicate information which is not representable using the standard, Kernel-supplied attributes, the communicating KSs need only agree on the name of a new attribute and the form of its value; this new attribute can then be used to pass the information. Thus, it is not necessary for a KS to Know the names of the other KSs involved. Individual KSs which

13 Each module is implemented as a separately compiled body of code. A configuration is specified at load time by selecting the desired modules. Additionally, any KS or precondition can be *inhibited* at run-time, effectively excising it from the system.

create, are activated by, or use this information may be added to or deleted from the system without requiring modifications to the other KSs.

A KS as a Hypothesis Generator

There are two major reasons, in addition to the one already discussed about context validation, why total independence was not achieved; both of these relate to a KS as a generator of hypotheses. The first reason concerns the control of the number of hypotheses a KS should initially generate and the reinvocation of it to generate additional, alternative hypotheses. The parameters associated with hypothesis generation should be set by a policy KS which has a more global view of the current state of the recognition process. The need then arises for a mechanism by which a policy KS can transmit its desires, in an anonymous and independent manner, to the appropriate KS.

It was hoped initially that these "processing goals" could be specified in terms of the basic hypothesize-and-test paradigm (i.e., by having the policy KS create the appropriate type of hypotheses which would in turn trigger the desired activity). However, "asking for something to be done" cannot always be specified conveniently in this way nor in an anonymous manner. For example, if there is a need for more word hypotheses to be generated in a particular time area, the action of creating a new hypothesis at the phrase level which will then be expanded at the word level does not precisely capture the desired activity, nor does the somewhat clumsy approach of modifying some attribute of the lower level data (e.g., the syllable level) to force a KS to reprocess this data so as to accomplish the desired activity. Note in this example, that by trying to force the concept of processing goal into the hypothesize-and-test paradigm, the policy KS must know the type of input stimulus that will trigger a KS to produce the desired results, thus violating the independence among modules. In addition, a KS which is designed to do hypothesizing-and-testing does not necessarily produce a response that will precisely match the desired processing goal. Due to these difficulties of directly embedding goal processing control in the hypothesize-and-test paradigm, an alternative approach was developed (but not implemented) which integrates smoothly with the data-directed control flow of Hearsay-11.

This alternative approach is based on introducing the concept of a g*oal node* into the blackboard, with types of attributes distinct from those of an hypothesis and a means of relating goals at different levels. The action of creating a goal at a particular level is a monitorable event that triggers a KS that can do processing at that level. By making a goal node distinct from an hypothesis, a policy KS can generate goals without interfering with KSs that operate at that information level but that cannot respond to the goal. If the triggered KS cannot directly satisfy the goal, it can generate a subgoal, linked to the original goal, to generate data at another level which could be used by the KS to satisfy the original goal. In this way, a policy KS can interact with KSs in an anonymous and independent way. For example, if there is no KS to react to the goal, processing can still continue. In the same manner, if there is more than one KS that can respond to the goal (i.e., competing KSs), the scheduler can resolve this conflict without the need for any action by the KS that generated the goal. A goal node can also be used as a convenient place for a generator type of KS action to leave internal state information about how much and what type of further processing it can do in this area.

The other major reason for violating the independence criterion was based on an efficiency consideration. As previously mentioned, it is comparatively expensive to create an hypothesis on the blackboard. The cost of hypothesis creation is especially critical with a KS that can potentially generate a large number of hypotheses. For example, the syntax prediction KS (EXTEND, in C2) can create, based on a prediction from a single phrase hypothesis, several hundred word hypotheses. Each of these must then be processed by the word verifier KS (MEW) and verified or rejected. Before these hypotheses are verified they share almost identical structures. All but twenty, perhaps, will be rejected by MEW. To avoid the expense of expanding these as distinct blackboard word hypotheses, special data structures have been constructed to store the predicted words compactly, these data structures are then attached as an attribute of the phrase hypothesis. This example further illustrates the weakness in the current Hearsay-II implementation of efficiently representing and processing groups of hypotheses.

Uniformity of Control

Another issue associated with the data-directed control structure is the ease with which different global control strategies can be explored. The uniform interface conventions used for specifying and activating KSs and preconditions, together with treating policy (strategy) KSs in the same way as other KSs makes the total system easy to modify and understand.

As part of the uniform convention for specifying each KS, non-procedural declarations are required which tell the system the type of pattern that triggers the KS and the type of action that can result from the activation of the KS. By separating the activation of a KS from its scheduling, it has been easy to introduce new global strategies by applying a new priority evaluation function to the information supplied by each KS. In addition, by allowing a policy KS to be able to trigger upon certain conditions that occur in the scheduling "data base" (such as the absence of any invoked KSs, or the lack of any invoked KSs above a certain priority level), it is possible to add different types of policy KSs into the system in a modular manner (e.g., WOSCTL in configuration C2).

In the initial specification of the Hearsay-II architecture, the approach required for focus of control was not well developed and represented one of the major conceptual problems which would determine the success of the design. As a result of work on this problem over the last three years, it is felt that the problem, though not completely solved, is now understood well enough so that it no longer represents a major obstacle to the effective use of the architecture. It is interesting to note that much of the discussion in preceding sections is based on a better understanding of what features need to be present in the architecture in order to efficiently support complex focus of control strategies.

Parallel Processing

One of the initial design goals of the Hearsay-II architecture was that it should be efficiently (and correctly) executable on a multiprocessor [Les75Pa and Fen75Mu]. In order to test the parallel processing capabilities of this architecture on an actual KS configuration, a multiprocessor simulation system was embedded in the multiprocess implementation of Hearsay-II. Each KS in this configuration was modified with the appropriate synchronization primitives.

The result of this simulation, which used an early version of the CI configuration that was strictly bottom-up in its processing (because it did not include the SASS module), showed that effective parallelism factors of four to six could be achieved [Fen77Pa]. Unfortunately, there does not exist similar simulation data for a fully configured CI or C2 configuration, both of which include top-down processing. However, it is expected that the C2 configuration would exhibit a much higher degree of parallelism, because KS interaction is more loosely-coupled and the system does a large amount of breadth-first type of search.

The parallelism factors of four to six that were achieved were less than expected. Further experiments were performed to determine the reason for these low factors. One of these experiments was to run the system with all uses of the synchronization primitives turned off. In this mode, the parallelism factors increased to fourteen. This dramatic increase is due to the fact that much superfluous synchronization was performed in each KS to maintain data consistency because no assumptions were made about how the blackboard was modified by other KSs. This superfluous synchronization, combined with synchronization primitives whose granularity of locking was too coarse, led to unnecessarily large areas of the blackboard being locked in order to maintain data consistency; this resulted both in significant interference among concurrently executing KS processes and a high system overhead (between 50 and 100 percent) in order to support parallel processing. As with context validation (discussed above), this was a price paid for complete independence among KSs.

A surprising result was that system performance, in terms of accuracy, was as good with the synchronization disabled as its performance with the full synchronization. The explanation for this phenomenon is that the asynchronous, data-directed control of Hearsay-II is robust in the face of certain types of synchronization errors. For example, consider the normal activity sequence of a KS which involves first examining the blackboard, and then, based on the values read, modifying the blackboard. Suppose that between the time when the KS read the value of an attribute on the blackboard and when it modified the blackboard, the value of the attribute was changed; therefore, the modification was inconsistent with the current state of the blackboard data. However, because of the data-directed nature of KS activation, the changing of the attribute will probably trigger the same KS to be reinvoked to recalculate its original modification. Thus, the need is obviated for a KS, while executing, to lockout the areas of the blackboard it has read, in order to maintain the consistency of its modifications. In addition, other types of inconsistency can often be resolved because another KS with a different view of the problem will correct an incorrect hypothesis whether it resulted from a synchronization error, a mistake in the theory used by the KS, or from errorful data. Thus, this self-correcting nature of information flow among KSs, created through the use of a data-directed form of the hypothesize-and-test paradigm, in many cases obviates the need for explicit use of synchronization.

CONCLUSIONS

The major conclusions on the use of the multi-level blackboard structure are the following:

1. The paradigm of viewing problem solving in terms of hypothesize-and-test actions distributed among distinct representations of the problem (where these representations form a hierarchy of abstractions) has been shown to be a computationally feasible approach to solving knowledge-intensive tasks. This paradigm also provides a convenient framework for structuring and applying knowledge. This has been demonstrated both by the successful application of the Hearsay-II architecture to the speech understanding task and also its adoption as an approach to problem-solving in a diverse set of other domains such as image understanding [Pra77Se], reading comprehension [Rum76To], protein-crystallographic analysis [Eng77Kn], signal understanding [Nii77Ru], and complex learning [Sol77Kn].
2. The representation of alternative hypotheses in an integrated manner on the blackboard has been shown to have positive aspects. In particular, the integrated representation avoids unnecessary recalculation and makes it easy to compute a global view of the current state of the problem solution, for the purpose of focussing. The problems still to be resolved arise because the integrated representation permits hypotheses to be used simultaneously in (shared by) multiple contexts (hypothesis networks). Existing primitives for grouping alternative hypotheses are inefficient in space, and, more importantly, make it difficult to determine easily the different contexts that use a hypothesis; these primitives also do not provide a convenient framework for representing and determining the fact that two contexts have very similar hypothesis structures.
3. There are problems with the current formulation of a partial solution as a distributed network of hypotheses at different information levels. There is a basic confusion in the Hearsay-II implementation between the hypothesis in the network which is at the highest level of abstraction (interpretation) and the entire network. This confusion, combined with the problem of handling of multiple uses of a hypothesis, makes it difficult to perform some of the complex focus-of-attention strategies possible in the architecture.
4. The uniform structure of the blackboard at all information levels has turned out to be a very positive feature of the architecture. It has made it possible to integrate new KSs into the system easily and to develop a large set of utilities applicable to all KSs. It has also permitted numerous reimplementations of the internal structure of the blackboard without requiring KS modification.

The major conclusions on the uniform, asynchronous, data-directed control structure are the following:

5. The use of an implicit and uniform control structure for KS cooperation makes the system easy to modify and understand. The separation permitted between the invocation of a KS and its scheduling makes it convenient to implement a variety of scheduling policies without KS modification.
6. The overhead costs involved in implementing this type of control structure are acceptable for KSs which do moderate amounts of internal computation at each invocation (e.g., more than 1/10 second in the current implementation).
7. This control structure is not appropriate for domains in which the hypothesis credibility ratings are not selective enough to suggest strongly good paths to search.
8. The problem of focus of attention in this type of control environment, though not completely solved, is now understood well enough so that it no longers represents a major obstacle to the effective use of the architecture. The integrated representation of alternatives on the blackboard, which permits a global view of the current state of problem solution, and the data-directed control structure make it possible to quickly refocus attention to the appropriate places in the blackboard.
9. The initial attempt to have complete KS independence (in both a sequential and parallel processing environment) resulted in a significant amount of overhead, and thus seems not to be worth the cost. A more balanced approach, based on some knowledge about the type of processing done by other KSs in the configuration, has been more effective. This knowledge does not violate anonymity of KSs because it is based on a functional characterization of their activity and not on their "names". Using this approach, KS configurations are still highly modular (i.e., there has been no serious subroutine interaction problem) without paying the severe costs (in complexity of KS programming and execution time) of complete independence.
10. Parallel processing can be exploited effectively in this architecture. The techniques which are needed because of the errorful nature of the processing in this problem domain provide a form of processing which is also robust in the face of data inconsistency caused by not imposing complete synchronization among parallel processes. Thus, the overhead costs of the synchronization are reduced substantially, allowing effective use of parallelism.

ACKNOWLEDGMENTS

Raj Reddy has provided much of the vision and energy for this work, most of the central ideas in the Hearsay model, and much technical expertise in many of the knowledge sources in the Hearsay-II system. Richard Fennell and Rick Hayes-Roth have been particularly instrumental in formulating and testing the Hearsay-II architecture. All members of the CMU "speech group" have contributed to this work; their substantial efforts are gratefully acknowledged. Oan Corkhill, Mark Fox, Doug Lenat, Jack Mostow,

14 Another example of this self-correcting type of computational structure is a class of iterative refinement methods used to solve partial differential equations. This type of computational structure can be decomposed for multiprocessor implementation so as to avoid most explicit synchronization at the expense of more cycles to reach convergence [Bau76As]. This decomposition is accomplished by not requiring each point in the differential grid to be calculated based on the most up-to-date value of its neighboring points.

Don McCracken, John McDermott, Allen Newell, Ed Riseman, and Elliot Soloway have made helpful suggestions for this paper.

APPENDIX - CONFIGURATIONS OF KNOWLEDGE SOURCES

Configuration CI

The KSs of CI (see Figure 1) are functionally described here briefly. The name given in parentheses following the name of the KS is the module in which it was embedded.

SEG (SEG) — The SEG KS [Gol76Se] generated, from the digitized acoustic signal, a sequence of contiguous, variable-length segment hypotheses.

CSEG (PSYN) -- This KS [Sho76Ph] combined segment hypotheses into larger segment hypotheses. The stimulus frame was a sequence of three contiguous segment hypotheses; the action was to generate one or more new segment hypotheses, each of whose times lay within the time span of the three hypotheses in the stimulus frame. The precondition for this KS was triggered highly asynchronously — whenever a new segment hypothesis was created. The KS was then invoked once for every pair of segment hypotheses immediately preceding and following the new one.

PSYN (PSYN) - This KS [Sho76Ph] created phone hypotheses, based on segment hypotheses. The stimulus frame was also a sequence of three contiguous segment hypotheses; the action was to generate one or more phonetic hypotheses, again with times within the boundaries of the stimulus hypotheses. The comment above about asynchrony of execution of CSEG also holds for PSYN.

POM (POMOW) - The POM KS [Smi76Wo] generated syllable hypotheses from phone hypotheses. The stimulus frame contained phone hypotheses that were classified as syllable nuclei; the action of the KS was to create syllable hypotheses based on the stimulus frame and adjacent segment hypotheses. The precondition for this KS was very complex because it made no assumptions about the order in which phone hypotheses would be created. Thus, the creation of a new phone hypothesis of any kind (syllable nucleus or other) triggered the precondition and caused an invocation of the KS for each nucleus hypothesis with which the new phone hypothesis might possibly interact.

MOW (POMOW) -- The MOW KS [Smi76Wo] generated word hypotheses from contiguous syllable hypotheses. The stimulus frame consisted of a newly-created syllable hypothesis; the output word hypotheses covered the same time as the stimulus hypothesis, but could also encompass syllable hypotheses on either side of the stimulus hypothesis (i.e., for multi-syllabic words.) If the stimulus hypothesis suggested a multi-syllabic word but the hypothesis for the other syllables did not exist, the word would not be hypothesized; however, if at some later time the required syllable hypothesis did appear, the KS would be triggered (by the new syllable) and the word hypothesized.

RECOG (SASS) -- This RECOGnition KS [Hay76Sy] used syntactic knowledge to generate phrase hypotheses from contiguous word or phrase hypotheses. The precondition triggered on a new phrase or word Hypothesis (or one with a changed rating). If the triggering hypothesis completed, with existing hypotheses, a phrase and the constituents were rated sufficiently high, the KS was invoked. This was a bottom-up parsing action.

PREDICT (SASS) - The PREDICTion KS [Hay76Sy] used syntactic knowledge to generate a new phrase hypothesis, given another phrase hypothesis that was highly rated. This was essentially a "sidewise" or "outward*' action.

RESPELL (SASS) - This KS [Hay76Sy], given a predicted phrase hypothesis (i.e., one with no links to lower level hypotheses, either phrase or word) with a sufficiently high prediction rating, generated hypotheses of the constituents (words and/or phrases) of the predicted hypothesis. Thus, respelling drove processing downward, from predicted hypotheses towards the word level, so that predictions could ultimately be matched to acoustic data and verified or rejected.

POSTDICT (SASS) — Given a weakly recognized or predicted phrase or word hypothesis, this KS [Hay76$y] looked for other hypotheses that tended to confirm it. Such hypotheses were linked to the "postdicted" hypothesis, increasing its rating.

WOM (WOMOS) -- This KS [Cro76Wo] was triggered on new word hypotheses that were not linked to syllable hypotheses (i.e., ones that were generated "from above", by RESPELL or PREDICT). For each such hypothesis, it generated (via a dictionary lookup) expected syllable hypotheses which were likely to describe it.

MPS (WOMOS) -- The MOS KS [Cro76Wo], given a new syllable hypothesis, generated (via a dictionary lookup) a set of surface-phonemic hypotheses which described the syllable.

TIME (POSSE) - This KS [Cro76Wo] responded to the creation of a new phone or surface-phonemic hypothesis and attempted to create a link between the new hypothesis and an existing hypothesis at the other level.

SEARCH (POSSE) - This KS [Cro76Wo] responded to the creation of a new link between a phone hypothesis and a surface-phoneme hypothesis and attempted to create new links adjacent to the triggering one. Thus, TIME and SEARCH together incrementally built, through structural connections on the blackboard, a synchronization of a sequence of surface-phonemes representing a syllable with a sequence of lower-level, acoustically-based phones. The SEARCH KS was very complex in that it built up competing synchronizations (multiple interpretations); this was done with localized, incremental actions and while attempting to have the competing interpretations share maximal consistent sub-structures.

RPQL (RPOL) — This policy KS [Hay76Hy] was responsible for propagating validity ratings. It triggered on the creation of an hypothesis, the establishment of a structural connection between two hypotheses, or the change of rating of an hypothesis. It calculated ratings for an hypothesis based on the values of KS-assigned attributes and the ratings of its structurally connected neighboring hypotheses.

FOCUS (FOCUS) —This policy KS imposed a global control strategy on the function of ail other KSs in the system. It imposed this control through the setting of goal hypotheses which indicated to a KS both that it should attempt to generate particular types of hypotheses and also what internal criterion (thresholds) it should apply in order to generate such hypotheses.

The strategy implemented by this KS was based on a progressive enlarging of the search space of hypotheses as existing hypotheses prove fruitless; the idea behind this strategy is that one should open up the combinatorics in the search space only when absolutely necessary. The strategy was implemented by setting up initial goal hypotheses with very high criteria for hypothesis generation and then successively lowering these thresholds when the search stagnated.

Configuration C2 (see Figure 2)

SEG (SEG) — Functionally similar to SEG in C1.

POM (POMOW) - This KS is similar to POM in C1, but hypothesizes directly from segment hypotheses (rather than phone hypotheses). Another difference from POM in CI is that the precondition here assumes that by the time it responds to the creation of a syllable nucleus segment hypothesis, all other segment hypotheses in the vicinity have also been created, thus simplifying the precondition considerably.

MOW (POMOW) -- Functionally similar to MOW in C1.

WORDCTL (WOSEQ) - This policy KS controls the generation of word hypotheses by MOW by creating "goal" hypotheses at each time area in the utterance which are interpreted by MOW as indicating how many word hypotheses are desirable in that area.

WOSEQ (WOSEQ) - This KS [Les77Se] uses pair-wise syntactic knowledge to create, from contiguous word hypotheses, word-sequence hypotheses. Two preconditions can invoke this KS: One precondition responds to the creation of new word hypotheses bottom-up (and typically fires once *per* utterance, after all such word hypotheses have been created). The other precondition is triggered by previously-created word-sequence hypotheses being marked "rejected".

WOSCTL (WOSEQ) — This policy KS monitors for stagnation in the search process; this condition is recognized if there *are* no

waiting KS instantiations above a certain priority or if the global measures of current state of the problem solution have not increased in the last n KS executions. If stagnation is recognized, this KS attempts to generate new word-sequence islands from which the search may be more fruitful. This is accomplished by decomposing existing word-sequence islands already on the blackboard or generating new islands which were initially discarded because their rating were too low.

PARSE (SASS) ~ This KS [Hay77Sy] uses the full constraints of the grammar to parse word-sequence hypotheses by searching a graph representation of the grammar. Each such parsing action is done internally to the KS; the parse trees themselves do not appear on the blackboard. The stimulus hypothesis is a newly-created word-sequence hypothesis. If the words do not parse, the stimulus hypothesis is marked as "rejected". If the words do parse (i.e., can occur contiguously in some sentence of the language defined by the grammar), a phrase hypothesis is created.

EXTEND (SASS) - The EXTEND KS [Hay77Sy] uses the grammar to predict words that might occur immediately preceding and following a phrase hypothesis. The stimulus hypothesis is a newly-created phrase hypothesis; the action is to a attach a "word-prediction" attribute to the hypothesis which names the predicted words.

MEW (POMOW) - The MEW KS is used to verify words which are predicted adjacent to a phrase hypothesis. The precondition triggers on a phrase hypothesis which has a word-prediction attribute added. An attempt is made to verify or reject each predicted word. The KS first checks the blackboard for a previously-hypothesized word that satisfies the time-adjacency criteria (i.e., immediately precedes/follows the predicting phrase). If none is found, a search is made of POMOW's internal store to see if the candidate can be matched by a word previously generated by MOW which has not been hypothesized on the blackboard. If one is still not found, the WIZARD procedure [McK77Wo] is called; this compares the segment hypotheses in the predicted area to a network description of possible pronunciations for the word. The result of the call to WIZARD is either a rejection of the predicted word, or else a verification, including a rating and an estimated end-time (or begin-time, if predicted preceding the phrase).[15] In general, several different "versions" of the word may be verified which differ in their end-times (begin-times); a word hypothesis is created for each such version and a "word-verification" attribute is added to the phrase hypothesis which names all the verified word hypotheses.

CONCAT (SASS) - The CONCAT KS [Hay77Sy] responds to a phrase hypothesis which has a word-verification attribute added. The action is, for each verified word, to parse the words of the original phrase augmented by the newly verified word. The extended phrase is then hypothesized. If all predictions to the right or left of the phrase are rejected, the phrase hypothesis is marked as rejected, as is the underlying word-sequence hypothesis. (Note that this last action will re-trigger WOSEQ to generate more word sequences.)

RPOL (RPOL) - This KS [Hay77Po] is similar to its counterpart in CI except that ratings of hypotheses above the word level (i.e., word sequences and phrases) are based on the word hypotheses that ultimately -underly them, rather than on the hypotheses that are directly connected to them from below (which are usually other word sequence or phrase hypotheses).

STOP (RPOL) - This policy KS [Mos77Ha] triggers on the creation of each phrase hypothesis whose initial and final supporting hypotheses are the unique "begin-" and "end-of-utterance" word hypotheses, respectively. Each such phrase hypothesis is a complete sentence and spans the entire utterance and thus is a candidate for selection as the system's recognition result. In general, the control and rating strategies do not guarantee that the first such complete spanning hypothesis found will have the highest rating of all poss.ble spanning sentence hypotheses that might be found if the search were allowed to continue, so the system should not just stop with the first one generated. However, the characteristics of such an hypothesis are used by STOP to prune from further consideration (by marking as "rejected") other partial hypotheses which, because of their low ratings, are unlikely to be extendible into spanning hypotheses with ratings higher than the best already-discovered spanning sentence. If the pruning process is severe enough, there will be no more partial phrase hypotheses left to consider by EXTEND; thus, KS activity will die out. This also triggers STOP, which then halts the system and selects the highest-rating complete spanning hypothesis as the "result". If this quiescence does not occur, the STOP KS will eventually force a halt after the expenditure of a predefined amount of computing resources (time or space). In this case, it also selects the highest-rated complete spanning hypothesis; if no such hypothesis exists, it selects several of the highest-rated partial phrase hypotheses as the "result". (This set of fragments is interpreted by the semantic interpretation program [Fox77Ma] which interfaces to the Hearsay-II system.)

15 WIZARD is, in effect, a miniature version of the HARPY speech recognition system [Low76Ha], except that it has one network for each word, rather than one network with all words and all sentences. The WIZARD procedure is also used in the MOW KS.

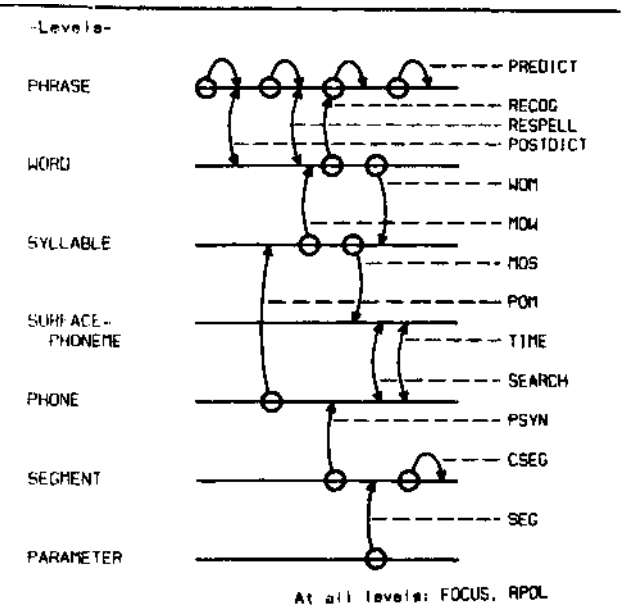
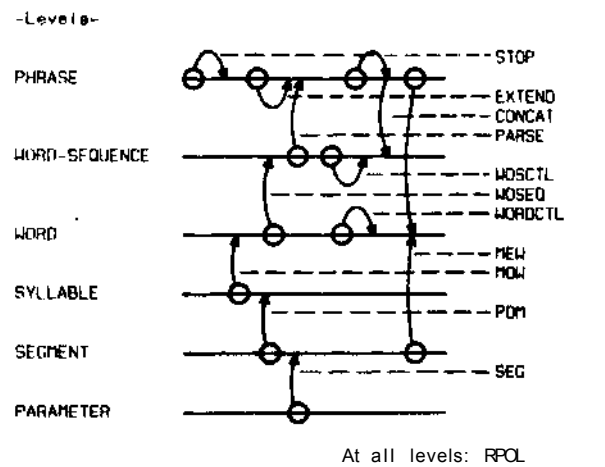


Figure 1: The levels and knowledge-sources of configuration C1. (As operational in January, 1976.)



Fiaury 2: The levels and Knowledge-sources of configuration C2. (As operational in September, 1976.)

REFERENCES

Bau76As Baudet, G. M. Asynchronous iterative methods for multiprocessors Tech. Rep., Dept, of Computer Science, Carnegie-Mellon Univ., Pittsburgh, Pa, Nov., 1976

Bar76Sc Barrow, H. G. and Tennenbaum, J M MSYS A system for reasoning about scenes Tach Rep 121, AI Canter, SRI, Menlo Park, Ca, Apr., 1976

CMU76W4 CMU Computer Science Speech Group Working papers *in* speech recognition IV: The Hearsay-II system Tech Rep, CMU, Feb, 1976

CMU77Su CMU Computer Science Speech Group Summary of the CMU Five-year ARPA effort in speech understanding research. Tech Rep, CMU, 1977

Cro76Wo Cronk, R and Erman, L D. Word verification in the Hearsay-II speech understanding system. Tech Rop, CMU, 1976

Eng77Kn Engetmore, R S and Nu, H P. A knowledge-based system for the interpretation of protein x-ray crystallogrnphic data. Tach. Rep Stan-CS-77-589, Stanford Univ, Stanford, CA Feb., 1977.

Erm7AEn Erman, L. D. An environment and system for machine understanding of connected speech Tech Rep, CMU, 1974 (PhD Dissertation, Comp Sei Dept, Stanford Univ.)

Erm75Mu Erman, I D and Lessor, V R A multi-level organization for problem solving using mony diveruo cooporating sources of knowledge. Proc 4th Inter, Joint Conf. on Artificial Intelligence, Tbilisi, USSR, 1975, 483-490.

Fen75Mu Fennoll, R D Multiprocess software architecture for AI problem solving Tech Rep, CMU, 1975 PhD Dissertation

Fen77Pa Fennell, R D and Leccor, V. R Parallelism in AI problem solving a case study of Hearsay-II IEEE Trons, on Computers. C-26 (Feb., 1977), 98-111

Fox77Ma Fox, M S and Mostow, D J Maximal Consistent Interpretations of Errorful Data in Hierarchically Modelled Domains Tech Rep, CMU, 1977.

Gol76Se Goldberg, H G Segmentation and labeling of connected speech Appeared in [CM76W4)

Hay76Hy Hayes-Roth, F, Erman, L D and Lesser, V R Hypothesis validity ratings in the Hearsay-II speech understanding system Appeared in (CM76W4]

Hey76Sy Hayes-Roth, F and Mostow, D J Syntax and semantics in a distributed speech understanding system Proc IEEE Inter Conf on Acoustics, Speech and Sip.nal Processing Philadelphia, PA, 1976, 421-424 Also appeared in [CMU76W4]

Hay77Fo Hayes-Roth, F and Lesser, R Focus of attention in the Hearsay-II system Tech Rep, CMU, 1977.

Hey77Po Hayes-Roth, F., Lesser, V R, Mostow, D J. and Erman, L. D. Policies for rating hypotheses, halting, and selecting a solution in Hearsay-II Appeared in [CMU77Su].

Hay77Ro Hayes-Roth, F The Role of Partial and Best Motchcs in Knowledge Systems The Rand Paper Series, P-5802, The Rand Corp, Santa Monica, Ca,Jan, 1977

Hay77Sy Hayes-Roth, F., Erman, L D, Fox, M and Mostow, D J Syntactic processing in Hearsay-II Appeared in [CMU77Su],

Hen75Ex Hendrix, G G Expanding the Utility of Semantic Networks Through Partitioning Prpc 4th IJCAI, Tbilisi, USSR, 1975, 115-121.

Hew72De Hewitt, C Description and theoretical analysis (using schemata) of Planner; A language for proving theorems and manipulating models in a robot AI Memo No 251, MIT Project MAC, 1972

Les750r Lesser, V R, Fennell, R D, Erman, L D and Reddy, D. R Organization of the Hearsay-II speech understanding system IEEE Trans, on ASSP 2.3 (Jan, 1975) 11-23

Lea75Pa Lesser, V R. Parallel processing in spoech understanding systems: a survey of design problems. In Spench Recognition Invited Papers of the IEEE Svmp (Reddy, D R, Ed) Academic Press, 1975, 481-499

Les77Se Lesser, V. R, Hayes-Roth, F, Birnbaum, M and Cronk, R Selection of word islands in the Hoarsay-II speech understanding system. Proc, 1977 IEEE Inter Conf on ASSP, Hartford, CT, 1977

Low76Ho Lowerre, B T Tho Harpy spoech recognition system Toch. Rep, CMU, 1976 PhD D.ssortation

McK77Wo McKeown, D M. Word verification in the Hearsay-II cpesch understanding system Proc 1977 IFEE Inter Conf. on ASSP, Hartford, Ct

Min74Fr Minsky, M A framework for representing knowledge. AI memo No. 306, MIT, June, 1974

Mos77Ha Mostow, D J A halting condition and related pruning houristic for combinatorial search Tech Rep, CMU, 1977

New73Sp Newell, A, Bamett, J, Forgio, J, Groon, C, KIntt, D., Licklider, J. C. R., Munson, J., Reddy, R and Woods, W. Speech Understanding Systemr Final Report of a Study Group North-Holland, 1973 (Originally appoared in 1971)

Nii77Ru Nn, H P and Fcigonbaum. E A Rulo-b.isod understanding of signals. Conf on Pattorn-Directed Inference Sygtomu, Hawaii, 1977.

Pra77So Prager, J, Nngin, P., Kohler, R, Hancen, A and Riseman, E. Sementation processos in tho VISIONS system IJCAI-77, Cambridge, Mass., 1977.

Red73Mo Reddy, D R, Erman, L D and Nroly, R B A model and a system for machine recognition of speech IEEE Trnns, on Audio and Electrocoustics, AU-21, 1973, 229-238.

Red73Hx Reddy, D R, Erman, L D., Fennoll, R D and Neely, R B The Hearsay speech understanding system an example of the recognition process. Proc 3rd IJCAI. Stanford, CA, 1973, 185-193

Rei76SA Reiser, J F SAIL Stanford Artificial Intelligence Laboratory, Memo AIM-289, 1976

Ros76Sc Rosenfeld, A, Hummel, R A and Zucker, S W Scene labeling by relaxation operations IEEE Trans Systems. Man and Cybernetics. SMC-6, 420-433, 1976

Rub77Lo Rubin, S M and Reddy, D R The LOCUS model of search and its use in image interpretation Proc 5th IJCAI. Cambridge, MA, 1977.

Rul73Qa Rulifson, J F, et al QA4 A procedural calculus for intuitive reasoning Tech Note 73, AI Center, SRI, Menlo Park, Ca, 1973.

Rum76To Rumelhert, D E. Toward an interactive model of reading Tech. Rep. 56, Center for Human Information Processing, Univ. of Cal. at San Diego, La Jolla, CA

Sho76Ph Shockey, L and Adam, C The phonetic component of the Hearsay-II speech understanding system In CM76W4.

Smi76Wo Smith, A R Word hypothesiration in the Hearsay-II speech system. Proc IEEE Int Conf on ASSP. Philadelphia, PA, 1976.

Sol77Kn Soloway, E M and Riseman, E M Knowledge-directed learning. Conf. on Pattern-Directed Inference Systems, Hawaii, 19Z7.

Wal770v Walker, D et al An overview of speoch understanding research at SRI Proc 5th IJCAI. Cambridge, MA, 1977.

Woo76Fi Woods, W A et al Speech understanding systems, Final report, Nov 74 - Oct 76 BBN report 3438, Bolt Beranek and Newman, Inc., Cambridge, MA, 1976.