System Organizations for Speech Understanding: Implications of Network and Multiprocessor Computer Architectures for AI

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

***Abstract*—This paper considers various factors affecting system organization for speech understanding research. The structure of the Hearsay system based on a set of cooperating, independent processes using the hypothesize-and-test paradigm is presented. Design considerations for the effective use of multiprocessor and network achitectures in speech understanding systems are presented: control of processes, interprocess communication and data sharing, resource allocation, and debugging are discussed.[1]**

***Index Terms*—Hardware for AI, multiprocessors, networks, parallel processing, real-time systems, software for AI, speech recognition, speech understanding, system organization.**

INTRODUCTION

SYSTEM organizations for speech understanding systems must address many problems: effective use of multiple sources of knowledge, anticipation and goal-direction in the analysis of the incoming utterance, real-time response, continuous monitoring of input device(s), errorful nature of the recognition process, exponential increase of processing requirements with the increase of desired accuracy, and so on. A particular model of speech perception [20] which attempts to solve the above problems involves the use of cooperating independent processes using a hypothesize-and-test paradigm. This paper examines the effect of the problem constraints and the model on system organizations, presents the structure of a system currently operational on a PDP-10 computer and discusses the implications of multiprocessor and network architectures.

Unlike many other problems in artificial intelligence (AI), speech understanding systems are characterized by the availability of diverse sources of knowledge, e.g., acoustic–phonetic rules, phonological rules, articulatory models of speech production, vocabulary and syntactic constraints, semantics of the task domain, user models, and so on. A major problem, then, is to develop paradigms which can make use of all the available sources of knowledge in the problem solution. At the same time, absence of one or more sources of knowledge should not cripple the system. Suppose each source of knowledge is represented within the system as a process. In order to remove or add sources of knowledge, each process must be independent, i.e., it must not require the presence of other processes in the system. But at the same time each process must cooperate with the other processes, i.e., it must be able to effectively use the information gathered by them about the incoming utterance. Thus, a major design step is to establish what information is to be shared among processes and how this information is to be communicated so as to maintain the independence of individual processes while still allowing for necessary process cooperation.

Knowledge available in the acoustic signal represents only one part of the total knowledge that is brought to bear in understanding a conversation. A good example of this is when one is interrupted by an appropriate response from the listener to a question that is as yet incomplete. In general, a human listener can tolerate a great deal of sloppiness and variability in speech because his knowledge base permits him to eliminate most of the possibilities even as he hears the first few words of the utterance (if not before!). We feel that this notion of anticipation, prediction, and hypothesis generation is essential for machine perception systems as well. In general, we expect every source of knowledge to be able to generate hypotheses in a given context, or verify hypotheses generated by others using different representations of knowledge, if necessary. The implication is that knowledge processes be organized within the system so as to reduce the problem of recognition and understanding to one of prediction and verification.

Manuscript received February 15, 1973. This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under Contract F44620-70-C-0107 and monitored by the Air Force Office of Scientific Research.

L. D. Erman, V. R. Lesser, and D. R. Reddy are with the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213.

R. D. Fennell was with the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213. He is now with the Federal Judicial Center, Washington, DC 20005.

[1] Since the writing of this paper (in 1973), we have made advances in solving many of the problems described here. These solutions are embodied in the Hearsay II system (see Lesser *et al.* [12], Fennell [9], Fennell and Lesser [10], Erman and Lesser [7], and Lesser [13]).

In tasks such as chess and theorem proving, the human has sufficient trouble himself so as to make reasonably crude computer programs of interest. But, because humans seem to perform effortlessly (and with only modest error) in speech (and visual) perception tasks, similar performance is expected from machines, i.e., one expects an immediate response and will not tolerate any errors. To equal human performance, a speech understanding system must be able to understand trivial questions as soon as they are uttered. This implies that various processes within the system should be allowed to operate as soon as there is sufficient incoming data, without waiting for the completion of the whole utterance. If the processes within the system are independent and unaware of the existence of each other, then the system must provide facilities for activation, termination, and resource allocation for each of the processes. Further, if a process can be deactivated before it reaches a natural termination point, provision must be made to preserve the state of the process until it is reactivated. Also, it is necessary to provide interlocks on the data that are shared among many processes.

This has several implications for system organization. The system must monitor the input device continuously to determine whether speech is present; this requires nontrivial processing. If the system is unable to process the incoming data, automatic buffering must be provided. If the system is to run on a time-sharing system, provision must be made to ensure that no data are lost because the program is swapped out for a period of time. If the speech understanding system is to consist of a set of cooperating independent processes, it is further necessary that they be able to be interrupted at unpreprogrammed points—if the microphone monitoring program is not activated in time to process the incoming utterance, it could lead to irrevocable loss of data. These considerations lead to two additional requirements that are not commonly available on existing time-sharing systems, viz., process-generated interrupts of other processes and user servicing of interrupts.

One of the characteristics of speech understanding systems is the presence of error at every level of analysis. To control such errors and permit recycling with improved definitions of the situation, one uses techniques such as feedforward, feedback, and probabalistic backtracking. If such facilities do not exist within the system, they must be programmed explicitly.

Speech, by its nature, appears to be computer intensive. A substantially unrestricted system capable of reliably understanding connected speech of many speakers using a large vocabulary is likely to require systems of the order of a proposed AI machine [3], i.e., processing power of 10 to 100 million instructions per second and memory of 100 to 1000 million bits.[2] To obtain such processing power, it appears necessary to consider multiprocessor architectures. Decomposition of speech processing systems to effectively use distributed processing power requires careful consideration even with primitive systems. Our model of cooperating independent processes, each representing a source of knowledge, leads to a natural decomposition of the algorithms for such machine architectures.

The Current Hearsay System

In this section we briefly describe the Hearsay speech understanding system as it now exists at Carnegie-Mellon University. (More detailed descriptions of the system are given in [20], [21], [6], [16].) We shall stress those aspects of its organization which are responsive to the constraints and model outlined above. This system represents a first attempt to solve those problems; thus, some of the constraints are only partially or poorly met, while others are satisfied in a more constricted way than necessary. We shall point out these limitations as they are described; later sections on closely coupled and loosely coupled processor network architectures describe possible corrections and improvements of the system.

The Hearsay system is implemented as a small number of parallel coroutines (see Fig. 1). Each coroutine (module) is realized as a separate job in the PDP-10 time-sharing system; thus the time-sharing monitor is the primary scheduler for the modules. In general, the modules may achieve a high degree of (pseudo) parallel activity (through the use of shared memory and a flexible interprocess message system[3]), but, in practice, we limit the parallelism to a very modest amount. This limitation is imposed for two reasons: first, since the PDP-10 is a uniprocessor system, there is nothing to be gained (in the time domain) by increasing the parallelism; and, second, the greater the amount of parallelism, the more difficult it is to control and debug the programs within a time-sharing system that is not designed for cooperating processes (jobs).

The model of recognition specifies that there be separate processes, each representing a different domain of knowledge. We have chosen three major domains of knowledge: acoustic–phonetics, syntax, and semantics.

1) The acoustic–phonetic domain, which we refer to as just *acoustics,* deals with the sounds of the language and how they relate to the speech signal produced by the speaker. This domain of knowledge has traditionally been the only one used in most previous attempts at speech recognition.

2) The *syntax* domain deals with the ordering of words in the utterance according to the grammar of the input language.

3) The *semantic* domain considers the meaning of the

[2] Smaller and substantially cheaper systems can be built to perform useful but restricted speech understanding tasks.

[3] The facilities provided for inter-job control and communication are similar to those developed for the Stanford Hand-Eye system [8].

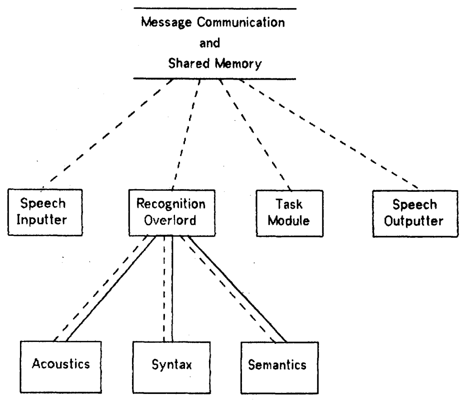

Fig. 1. Decomposition of processes in the current Hearsay system.

utterances of the language, in the context of the task that is specified for the speech understanding system.

These processes, according to the model, are to be independent and removable; therefore the functioning (and very existence) of each must not be necessary or crucial to the others. On the other hand, the model also requires that the processes cooperate and that the recognition should run efficiently and with good error recovery; these dictates imply that there be a great deal of interaction among the processes. Thus we seem to have opposing requirements for the system. These opposing requirements led to the design of the following structure:

> Each process interfaces externally in a uniform way that is identical across processes; no process knows what or how many other recognition processes exist.
>
> A mediator, ROVER (Recognition OVERlord), handles the interface to each of the processes and thus serves as the linkage connecting the processes; the processes are called ROVER's "sons."

The interface is implemented as a global data structure which is maintained by ROVER. Each of ROVER's sons puts information into this data structure in a uniform way. Each may access information submitted by its brothers, but in a manner which leaves the source of that information anonymous. This mechanism is analogous to a bulletin board on which messages can be left by several people and for which there is a monitor who accepts the message and arranges them in appropriate places on the board for others to react.

This anonymous interface structure is appropriate only if the global data structure can be designed in such a way as to allow the processes to communicate meaningfully, i.e., there must be a common language which allows them to transmit the kind of information they need to help each other to work on the problem. We resolve this problem by using the *word* as the basic unit of discourse among the processes.

The basic element of the global data structure is the *word hypothesis* which represents an assertion that a particular word (of the input language lexicon) occurs in a specified position in the spoken input. A *sentence hypothesis* is an ordered linear sequence of word hypotheses; it represents an assertion that the words occur in the sentence in the order that the word hypotheses appear in the sentence hypothesis. In addition, the unique "word" FILLER may appear as a word hypothesis; this is a placeholder and represents the assertion that zero or more as yet unspecified words occur in this position in the spoken sentence. In general, there may be any number of sentence hypotheses existing at any one time.

The interactions among the source-of-knowledge processes are carried out using the hypothesize-and-test paradigm prescribed by the model. In general, any process may make a set of hypotheses about the utterance; all the processes (including the hypothesizer) may then verify (i.e. reject, accept, or reorder) these hypotheses. In particular, hypothesization occurs when a recognition process (acoustics, syntax, or semantics) chooses a FILLER word from a sentence hypothesis and associates with it one or more *option words,* each of which it asserts is a candidate to replace all or part of the FILLER. Verification consists of each process examining the option words and rating them in the context of the rest of the sentence hypothesis.

Several restrictions have been placed on the implementation of this general scheme. First, at any time only one part of the shared, global data structure (i.e., one sentence hypothesis) is accessible to the processes for hypothesization and verification. Second, the processes go through the hypothesization and verification stages (and several other subsidiary stages) in a synchronized and noninterruptable manner. Finally, only one process is allowed to hypothesize at any one time. Again, these restrictions were imposed both because parallelism on a uniprocessor does not accomplish any throughput increase and because the available programming and operating systems make a more general implementation difficult to specify, debug, and instrument.

These restrictions are mitigated somewhat by carefully adjusting the time grain of the processing so that each noninterruptable phase is not "excessively large."

Each sentence hypothesis has a confidence rating associated with it which is an estimate of how well it describes the spoken utterance. This rating is calculated by ROVER, based on information supplied by the recognition processes. Errors in processing become evident when the overall rating given to a sentence hypothesis begins to drop; at that point, attention is focused on some other sentence hypothesis with a higher rating. This switching of focus is the mechanism that provides the error recovery and backtracking that is necessary in any speech understanding system.

Closely Coupled Processor System Organizations

As discussed in the Introduction, in order to do real-time speech understanding a substantial amount of computing power is required. Recent trends in technology indicate that this computing power can be economically obtained through a closely coupled network of "simple" processors, where these processors can be interconnected to communicate in a variety of ways (e.g., directly with each other through a highly multiplexed switch connected to a large shared memory [2] or through a regular or irregular network of busses [4]. However, the major problem with this network approach to generating computing power is finding algorithms which have the appropriate control and data structures for exploiting the parallelism available in the network. The model for a speech understanding system as previously discussed, which is decomposed into a set of independent processes cooperating through a hypothesize-and-test paradigm, represents a natural structure for exploiting this network parallelism.

There exist three major areas for exploitation of parallelism in the structure of this speech understanding system: preprocessing, hypothesization and verification, and the processing specific to each source of knowledge. The *preprocessing task* involves the repetition of a sequence of simple transformations on the acoustic data, e.g., detection of the beginning and end of speech, amplitude normalization, a simple phoneme-like labeling, smoothing, etc. This sequence of transformations can be structured as a pipeline computation in which each transformation is a stage in the pipe. Thus, through this pipeline decomposition of the preprocessing task, a limited amount (i.e., 4) of parallel activity is generated.

The *hypothesize-and-test paradigm* for sequencing the activity of the different sources of knowledge can also be structured so as to exhibit parallelism, but the amount of parallelism is potentially much greater. This parallel activity is generated by the simultaneous processing of multiple sentence hypotheses and the simultaneous hypothesization and verification by all sources of knowledge. The simultaneous processing of multiple sentence hypotheses, rather than processing just the currently most likely candidate, can conceptually introduce unnecessary work. But in practice, because of the errorful nature of the processing, there may be a considerable amount of necessary backtracking to find the best matching sentence hypothesis. It is appropriate to quote a conjecture of Minsky and Papert [15, sect. 12.7.6] on this point.

> [While for the exact match problem] relatively small factors of redundancy in memory size yield very large increases in speed, . . . [for the best match problem]. . . . for large data sets with long word lengths there are no practical alternatives to large searches that inspect large parts of the memory.]

Thus, the parallel activity generated by simultaneous processing of more than one sentence hypothesis can result in a proportional speed-up of the recognition process.[4] Correspondingly, simultaneous hypothesization and verification by all sources of knowledge also results in a proportional speed-up of the recognition process because each source of knowledge is independent and is designed so that its knowledge contribution is additive.

Finally, the *verification algorithm* of each source of knowledge can be decomposed into a set of parallel processes in two ways. The first kind of decomposition is based on the fact that verifications are performed on a set of option words rather than a single word at a time. Thus, for each source of knowledge there can be multiple instantiations of its verification process, each operating on a different option word. The second kind of decomposition involves the parallelizing of the verification algorithms themselves; thus, each instantiation of a verification process may itself be composed of a set of parallel processes. However, this set of instantiations may not be totally independent because the rating produced by the verification process may be dependent on the particular set of option words to be verified and also on the local data base which is common to all the instantiations. For example, the acoustic verification process is a hierarchical series of progressively more sophisticated tests. The first few levels of testing look only at the context of a single option word, while the more sophisticated tests compare one option word against another. Thus, only at the first few levels of tests can the acoustic verification algorithm be parallelized in a straightforward manner.

The parallelism generated by parallelizing the hypothesize-and-test control structure and the verification processes are multiplicative in their parallel activity (i.e., performing in parallel the updating of n sentence hypothesis where each hypothesis invokes m verification processes and each verification process operates on o option words leads to a potential parallelism of $n*m*o$). This parallelism, together with the pipeline

[4] Simulation studies are currently being carried out on evaluating this speed-up factor. These studies are based on data generated from the current version of the Hearsay system.

parallelism of the preprocessing, leads to what appears to be a large amount of potential parallelism to be exploited by a closely coupled network. However, it is still not clear just how much potential parallel activity exists over the entire recognition system; nor is it known how much of this potential will be dissipated because of software and hardware overhead.

In order to answer these questions, a parallel decomposition of the Hearsay speech understanding system is now being implemented on C.mmp, a closely coupled network of PDP-11's which communicate through a large shared memory [2]. The C.mmp hardware configuration can contain up to 16 PDP-11's; the highly multiplexed switch that connects processors to memory permits up to 16 simultaneous memory references if these references are not to the same memory module. Thus, if processors are referencing different memory modules, then each processor can run at full speed. In addition, C.mmp can be configured for a specific application (e.g., speech) by replacing a processor by a special purpose hardware device which directly accesses memory (e.g., a signal processor).

The Hydra software operating system [24], which is associated with C.mmp, provides an appropriate kernel set of facilities for implementing the parallel version of the speech system. These facilities permit control of real-time devices, convenient building of a tree of processes, message queues, and shared data base communication among processes, user-defined scheduling strategies, arbitrary interruption of running processes, and dynamic creation of new processes. Building up from this base, a debugging system will be constructed which, in addition to the normal features, will permit the recording of all communication among processes, the tracing of all process activity, and the monitoring of global variables (including a recording of which processes have modified them). These additional capabilities are crucial for isolating errors and understanding the dynamic behavior patterns of the parallel system.

The major software problem to be investigated in this parallel implementation of the Hearsay system is how to efficiently map virtual parallelism (process activity) into actual parallelism (processor activity). This mapping problem in turn centers on three design issues, each of which relates to how processes interact:

1) the design of the interlock structure for a shared data base;
2) the choice of the smallest computational grain at which the system exhibits parallel activity; and
3) the techniques for scheduling a large number of closely coupled processes.

The first design issue is important because in a closely coupled process structure many processes may attempt to access a shared data base at the same time. In a uniprocessor system, the sequentialization of access to this shared data base does not significantly affect performance because there is only one process running at a time. In a multiprocessor system, however, if the interlock structure for a shared data base is not properly designed so as to permit as many noninterfering accesses as possible, then access to the shared data base becomes a significant bottleneck in the system's performance [14].

The second issue relates to how closely coupled processes can interact. If the grain of decomposition is such that the overhead involved in process communication is significant in relation to the amount of computation done by the process, then the added virtual parallelism achieved by a finer decomposition can decrease, rather than increase, the performance of the system. Thus, understanding the relationship between the grain of decomposition and the overhead of communication is an important design parameter.

The third issue relates to a phenomenon called the "control working set" [11]. This phenomenon predicts that the execution of a closely coupled process structure on a multiprocessor may result in a significant amount of supervisory overhead caused by a large number of process context switches. The reason for this high number of process context switches is analogous to the reason for "thrashing" within a data working set [5]. For example, in a uniprocessor system, if two parallel processes closely interact with each other, then each time one process is waiting for a communication from the other it would have to be context switched so as to allow the other process to execute. If these two processes communicate often, then there would be a large number of context switches. However, if there were two processors, each containing one of the processes, then there would be no process switching.

The implications of this phenomenon on constructing process structures are the following:

1) Processes should be formed into clusters where communication among cluster members is closely coupled whereas communication among clusters is loosely coupled. This process structuring paradigm has also been suggested as a model for the operation of complex human and natural systems [22].

2) The size of a process cluster cannot be chosen independent of the particular hardware configuration that will be used to execute it. For example, a cluster size of 8 may be appropriate for a hardware system containing 16 processors while being inappropriate for a system containing 6 processors.

3) The scheduler of a multiprocessor system should use a strategy that schedules process clusters rather than single processes. (This is analogous to the advantage of preloading the data working set rather than dynamically constructing the working set at each context swap.)

4) The use of process structures to implement inherently sequential, though complex, control structures (e.g., coroutines, etc.) may lead to inefficient scheduling of process structures on a multiprocessor system (i.e., the scheduling strategy should be able to easily differ-

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.