PASS: A Parallel Speech Understanding System

Sang-Hwa Chung, Ronald F. DeMara, and Dan I. Moldovan

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-2562

Abstract

We present a parallel computational model for the integration of speech and natural language processing. The model adopts a hierarchically-structured knowledge base and memory-based parsing techniques. Processing is carried out by passing multiple markers in parallel through the knowledge base. Speech-specific problems such as insertion, deletion, substitution, and word boundaries have been analyzed and their parallel solutions are provided. The complete system has been implemented on a parallel machine and is operational. Results show an 80% sentence recognition rate for the Air Traffic Control domain. Moreover, a 10-fold speed-up can be obtained over an identical sequential implementation with an increasing speed advantage as the size of the knowledge base grows.

1 Introduction

A key issue in spoken language processing has become the integration of speech understanding and natural language processing (NLP). Their effective integration offers higher potential recognition rates than recognition using word-level knowledge alone. Without higher level knowledge, error rates for even the best currently available systems are fairly high. For example, CMU's Sphinx system [7] is considered to be one of the best speaker-independent continuous-speech recognition systems today. But even the Sphinx system has a word recognition accuracy of only 70.6% for speaker-independent, continuous-speech recognition with a vocabulary of 1000 words, when recognizing individual words in sentences without using syntax or semantic information [7].

Clearly, we need some forms of high-level knowledge to better understand continuous speech. The integration of speech and NLP resolves multiple ambiguous hypotheses using syntactic, semantic and contextual knowledge sources. Since this requires sizable computation involving multiple levels of knowledge sources, speed performance degrades on realistic knowledge bases suitable for broader, complex domains.

The high volume of computational requirement of integrated speech understanding algorithms calls for new approaches to parallel processing. Recent work on parallel parsing is relatively limited. Moreover, most of the works relate to written language. Huang and Guthrie [6] proposed a parallel model for natural language parsing based on a combined syntax and semantics approach. Waltz and Pollack [9] investigated parallel approaches under paradigms related to the connectionist model. Giachin and Rullent [5] implemented a parallel parser for spoken natural language on a Transputer-based distributed architecture. They used a case frame-based parsing scheme and reported a sentence recognition accuracy of about 80% from continuously-uttered sentences, on average 7 words long, with a dictionary of 1000 words.

In this paper, we describe how the scalability problem can be addressed in the integrated PArallel Speech understanding System called *PASS*. A memory-based parsing model and parallel marker-passing schemes form the underlying philosophy of the system. We show how to handle the major speech-specific problems such as insertion, deletion, substitution, and word boundaries using tightly-coupled iterations between low-level phoneme sequences and higher-level concepts. We describe an operational implementation of our integrated approach and analyze its performance on a real parallel machine.

2 Integrated Speech and Natural Language Processing

The objective of PASS is to provide a parallel system which integrates speech and high-level NLP to improve processing speed and tractable domain size.

1043-0989/93 $03.00 © 1993 IEEE

2.1 System Overview

PASS contains the natural language understanding (NLU) module, the speech understanding (SU) module, and the knowledge base. The inputs to PASS are provided by the Phonetic Engine [8] manufactured by Speech Systems Incorporated. The Phonetic Engine provides a stream of phonetic codes for processing. It can handle speaker-independent continuous speech input with a large vocabulary of up to 40,000 words in real-time.

The phonetic codes provided by the Phonetic Engine are evaluated in the SU module to find the matching phoneme sequences. The predictions provided by the NLU module help the SU module to handle multiple hypotheses efficiently and reduce the search space. Word candidates activated by the SU module are further evaluated in the NLU module to construct meaning representations and generate a sentence output. The predictions and activations are performed in parallel by markers throughout the knowledge base.

The knowledge base for PASS is composed of concept sequences and phoneme sequences which are organized hierarchically. A *concept sequence* is a basic building block of sentences in a memory-based parsing scheme. In each concept sequence, concept nodes are connected by *first*, *next*, and *last* links. Similarly, in each *phoneme sequence*, phonemes are connected by *pfirst*, *pnext*, and *plast* links. Concept sequences are stored hierarchically in the memory network. In other words, more general concept sequences are placed at higher levels, and more specific concept sequences are placed at lower levels. This hierarchy is called a *concept sequence hierarchy*. Phoneme sequences exist at the lowest level of the concept sequence hierarchy attached to the corresponding concept nodes.

Currently, PASS works in the Air Traffic Control (ATC) domain developed by Speech Systems Incorporated for training air traffic controllers. ''`Tiger six sixteen, reduce speed to one five zero`'' is a typical target sentence in the ATC domain. The ATC domain contains a vocabulary of approximately 200 words and a semantic network of approximately 1400 nodes.

2.2 The Alignment Scoring Model

One of the early tasks to perform in speech understanding is to find the best matches of an input sequence of phonetic codes with phoneme sequences. To evaluate the matches, we use a codebook which was derived during the development process using automatically labeled speech data. The codes represent acoustic events having some ambiguity with respect to phonemes[1]. That is, two or more successive phonemes may be time-aligned with a single code and two or more successive codes may be time-aligned with a single phoneme.

To compute an alignment score between the phonetic code sequence S and the sentence transcription T, we first assign score values to the time-aligned subsequences of S and T, and then take the sum of these score values. To compute the score of the time-aligned subsequences, we account for: 1) each alignment between a code and a phoneme, 2) the number of successive phonemes aligned with the same code, and 3) the number of successive codes aligned with the same phoneme. To express the score of an alignment, we introduce three matrices:

- *X(code, phoneme)* - each element x_{ij} is the score to align *code* i with *phoneme* j. The X matrix is generally known as a *confusion matrix*.
- *Y(code, #phoneme)* - each element y_{ij} is the score to align *code* i with number(j) of successive phonemes.
- *Z(phoneme, #code)* - each element z_{ij} is the score to align *phoneme* i with number(j) of successive codes.

The score of an entire utterance is simply the sum of the scores of the time-aligned subsequences of the entire utterance. Some of the speech-specific problems like insertion, deletion and substitution are handled nicely by the alignment scoring model previously mentioned [1].

2.3 The Functional Structure of PASS

The architecture of the integrated memory-based speech understanding system is shown in Figure 1. Dark arrows indicate execution flow and light arrows show information flow. The SU module performs: *phoneme prediction, phoneme activation, word boundary control, oversegmentation control,* and *undersegmentation control.* The NLU module performs: *word prediction, word activation, multiple hypotheses control, meaning representation construction,* and *sentence generation.* The knowledge base containing concept sequences and phoneme sequences is distributed to each parallel processing element and the scoring information including X, Y and Z matrices is in a host or central controller.

In principle, the parallel speech understanding algorithm is based on a combination of top-down prediction and bottom-up activation in the concept sequence

[1] Our system actually has 1644 different phonetic codes generated by the Phonetic Engine while only 49 phonemes are used to represent phoneme sequences. The codes are represented by integers between 0 and 1643.

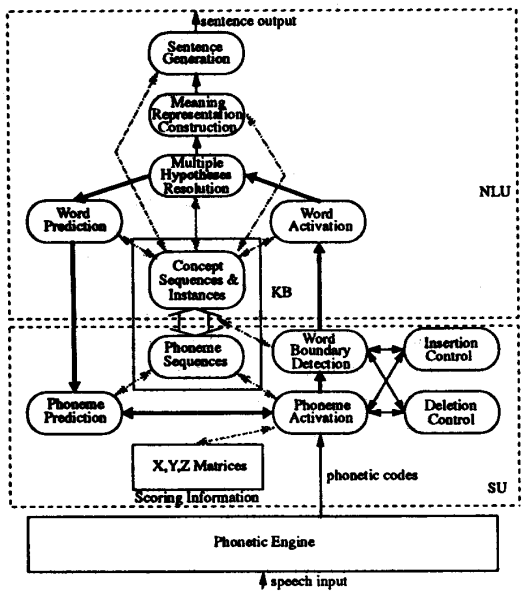

Figure 1: Modules within PASS.

hierarchy. In other words, top-down prediction decides the candidates to be evaluated next and bottom-up recognition activates a set of phonemes from a given phonetic code.

As shown in Figure 1, a circular path exists between the NLU module and the SU module as follows: word prediction → phoneme prediction → phoneme activation → word boundary control → word activation → multiple hypotheses control → word prediction. The operation starts in the NLU module by predicting the first words in all concept sequences. This in turn impacts the prediction of the first phonemes for these predicted words. Next, the system accepts a phonetic code as speech input, and via X, Y, Z matrices all the relevant phonemes are activated. The candidates of predicted and activated phonemes trigger further phoneme predictions. This process repeats until new word hypotheses are formed. This coincides with the activation of corresponding words, and the process is moved to the NLU module. Here, through a process similar to the one at the SU level, only the coincidence of predicted and activated words triggers further word predictions. When a word gets both prediction and activation, a concept instance is generated to construct a meaning representation. (For the details of meaning representation construction, refer to [2]). All words originally predicted, but not activated, receive cancellation markers. Afterwards, this cycle repeats.

The repeated top-down prediction and bottom-up activation are performed on this circular path until all phonetic code inputs are processed. The actual predictions and activations are performed by propagating markers in parallel through the knowledge base.

The processing of speech input on PASS requires the creation and movement of markers on the memory network. The actual implementation required about 20 different types of markers, including both *fat-markers* and simple *bit-markers*. Fat-markers carry scoring information and move around the memory network by marker propagation operations, while bit-markers only represent certain characteristics of nodes and are not propagated. Some fat-markers performing important functions are:

- *P-Markers* - indicate the next possible nodes (or phonemes) to be activated in the concept sequence (or phoneme sequence). They are initially placed on the first nodes of concept sequences and phoneme sequences, and move through the *next* (or *pnext*) link.
- *A-Markers* - indicate activations of nodes. They propagate upward through the concept sequence hierarchy.
- *I-Markers* - indicate instantiations of activated nodes. Activated concept nodes are finally identified by I-Markers.
- *C-Markers* - indicate cancellations of activated nodes. Because of multiple hypotheses, some I-Markers may be cancelled or invalidated later as their scores become inferior.

2.4 Marker-passing Solutions for Speech-specific Problems

During the phoneme sequence recognition process, the insertion and deletion problems occur in almost all words. It is critical to handle these problems properly for successful recognition of phonemes. The alignment scoring model provides the necessary information to solve the problems by the X,Y, and Z matrices.

However, it is not easy to simultaneously align a sequence of phonetic codes containing such problems with multiple phoneme sequence targets in the memory network, because the multiple targets should be evaluated at the same time, and the next phoneme activations cannot be foreseen while the current phonetic code is being processed. Therefore, when we advance the prediction markers based on the current scoring information, we should consider how to recover from bad expectations in some phoneme sequence candidates, without affecting good expectations in other candidates.

An example describing these problems is shown in Figure 2, where only a part of the concept sequence

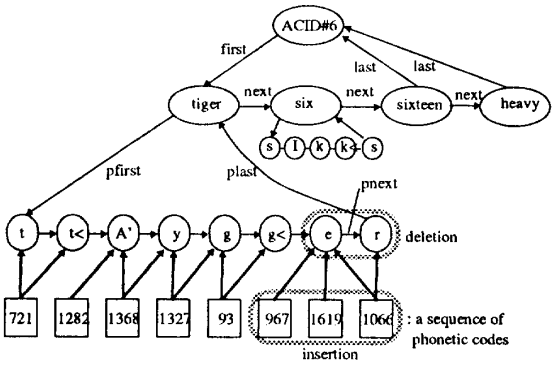

Figure 2: An Example of Handling the Insertion and Deletion Problems.

hierarchy for the ATC domain is depicted. In this figure, we show the time alignment between the subsequence of phonetic codes: 721 1282 1368 1327 93 967 1619 1066, and the phoneme sequence for the word *tiger*: t t< A' y g g< e r.

As an example, let us assume that we have processed the subsequence of codes: 721 1282 1368 1327 93. The remaining codes can be handled as follows (dual prediction markers, P(-1) and P(0), are used to keep the previously used P-Marker as well as the current P-Marker):

1. Code *967* is now accepted from the Phonetic Engine. By consulting the X matrix in the central controller, phoneme *e* is activated. The scoring process is as follows: Score(P(0)) = Previous_Score(P(0)) + Score(A), where Score(A) = $X(967, e) + Y(967, 1) + Z(e, 1)$. Then, P(0) is propagated to phoneme *r* from phoneme *e* through the *pnext* link. Phoneme *e* also keeps P(-1) to prepare against a possible insertion.

2. When code *1619* arrives, phoneme *e* is activated again, instead of phoneme *r*. That is, an insertion exists. The insertion handling routine calculates the score of the A-Marker: Score(A) = $X(1619, e) + Y(1619, 1) + Z(e, 2)$ where the previous $Z(e, 1)$ need not be cancelled because scores in the Y, Z matrices only contain offset values to avoid unnecessary computations. After adding the score of A to P(-1), a new P(0) is propagated again to phoneme *r*.

3. When code *1066* arrives, phoneme *e* and phoneme *r* are activated together. That is, both an insertion and a deletion occur at the same time. By this, we can assume that there exist no more insertions to phoneme *e*. Now, the score of the A-Marker is calculated as: Score(A) = $X(1066, e) + Y(1066, 1) + Z(e, 3)$. Again, a new P(0) adding the score of A is propagated to phoneme *r*. Here, a collision between P(0) and A exists, and the scoring process for phoneme *r* begins. Because phoneme *r* is the last phoneme activated in the phoneme sequence, an A-Marker propagates upward through the concept sequence hierarchy, and finally a new P(0) arrives at the first phoneme of the phoneme sequence for concept *six*.

Because multiple hypotheses are evaluated at the same time, recognition processes similar to the one described above are performed in parallel throughout the memory network. Substitution problems are not shown in Figure 2. When a substitution occurs, the score of the X Matrix for the substituted code-phoneme pair will be low or even below a threshold. Thus, when a substitution problem occurs in a phoneme sequence candidate, the score of the candidate is decreased. This candidate may be rejected when other hypotheses get better scores in any decision point.

2.5 Multiple Hypotheses Resolution

Multiple hypotheses cannot be completely resolved with the information available at the phoneme sequence level. The SU module activates multiple competing words, or even the same words repeatedly when insertion problems exist. When A-Markers are propagated up through the concept sequence hierarchy at a word boundary, several candidates may exist at any merging point. A merging point exists when a concept node contains multiple incoming *last* links, *isa* links, or *next* links. A merging point also exists when a concept node has multiple *plast* links coming from multiple phoneme sequences representing various pronunciations of the concept node.

In PASS, a scheme for score-based multiple hypotheses resolution is performed in parallel using marker-passing. When multiple candidates arrive at a merging point, the concept node in the merging point might have been either already activated from the evaluations of the previous input codes or first visited this time. To get the best candidate, the scores carried by the candidates' A-Markers are compared. If the concept node was activated previously, the score of the I-Marker is also compared with the scores of the A-Markers. The candidates with lower scores are cancelled by propagating C-Markers down through the

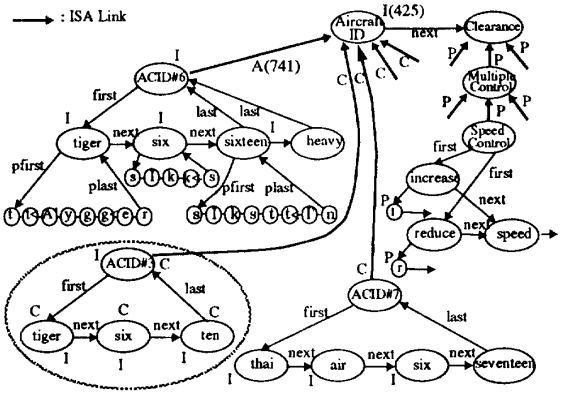

Figure 3: An Example of Multiple Hypotheses Resolution.

concept sequence hierarchy. As a result, the concept node gets a new I-Marker containing the highest score.

An example for the multiple hypotheses resolution is shown in Figure 3. The concept node *Aircraft ID* was previously activated by the hypothesis: **tiger six ten**, and the score of the I-Marker for the node is 425. Because this hypothesis was not a correct one, the activation score was poor. Although **tiger six ten** is apparently different from **tiger six sixteen**, it is still possible to activate this hypothesis with a low score. Basically, any phoneme with the X Matrix score greater than a certain threshold can be activated from an input code regardless of its meaning. So, it is critical to set the threshold properly to prevent unnecessary activations without losing any meaningful information.

When a new hypothesis: **tiger six sixteen** arrives at the node *Aircraft ID* with the score of the A-Marker, 741, the previous hypothesis is rejected because of its lower score. C-Markers are parallelly propagated down through all possible links in the concept sequence hierarchy except the link to the newly selected hypothesis. When C-Markers collide with I-Markers during the propagation, the I-Markers are cancelled. To protect partially evaluated hypotheses, a C-Marker in each node stops its propagation when the node does not contain an I-Marker. For example, **thai air six seventeen** is still in the middle of evaluation as shown, while later, turning out to be the hypothesis with the highest score. Thus, the I-Markers on the nodes *thai*, *air* and *six* need to be retained.

After resolving multiple hypotheses, a P-Marker is propagated to the concept node *Clearance* through the *next* link. From the node *Clearance*, P-Markers are propagated down to the first phonemes of the next possible phoneme sequences as shown. The activation, cancellation, and prediction operations are performed simultaneously for all possible hypotheses.

3 Execution Results

The PASS integrated speech/NLP algorithm has been implemented on the SNAP-1 parallel machine and is operational. Currently, the Air Traffic Control domain is being used and a larger Radiology domain is in development. We describe below the current system configuration and its performance.

3.1 SNAP-1 Parallel Processing Platform

SNAP-1 is a parallel array processor designed for semantic network processing with a reasoning mechanism based on marker-passing [4]. The SNAP-1 architecture is based on an a multiprocessing array and a dual-processor array controller. The array stores a semantic network of up to 32K nodes and 320K links. The processor array consists of 144 Texas Instruments TMS320C30 DSP microprocessors. The array is organized as 32 tightly-coupled *clusters* of 4 to 5 Processing Elements (PEs) each[2]. Each cluster manages 1024 semantic network nodes.

The central controller interfaces the processor array with a SUN4/280 host where application programs are written and compiled. Within the SNAP-C environment, high-level operations and libraries are provided for marker-passing. The parallel operations are initiated through a global bus from the controller which begins each propagation cycle by broadcasting marker instructions to the array. However, most of the work is performed locally though the propagation of markers at nodes within the cluster. Thus, several instructions and multiple propagations can be performed simultaneously.

3.2 System Performance

We have analyzed the execution of PASS for the ATC domain on the SNAP-1 platform and a uniprocessor. The essential elements of the domain, including basic concept sequences and phoneme sequences occupy 1,357 semantic network nodes with 5,834 links. Results are reported for a 16 cluster configuration which was the maximum available at the time. The

[2]Presently, 16 clusters are implemented in the full 5 PE configuration while the remaining 16 clusters have 4 PEs each, totaling 144 PEs.

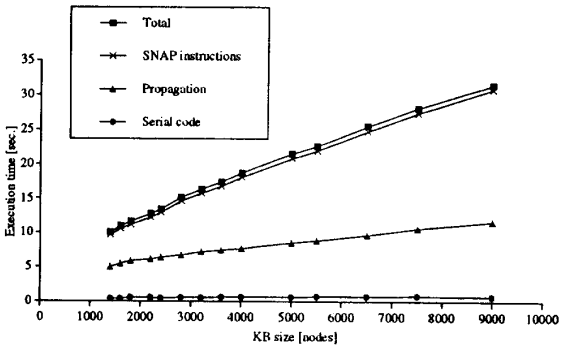

Figure 4: Application Size-up.

controller clock speed was 32 MHz while the array processors operated at 25 MHz.

We tested 60 different continuously-uttered sentences. Each sentence contained about 11 words and was spoken by four different speakers. An 80% sentence recognition rate was achieved within the domain for these untrained speakers. However, the majority of sentence-level failures had nearly correct semantic meaning representations. Thus, we anticipate that the sentence recognition rate can be further improved with additional high-level information.

Knowledge Base Scale-up

The size of the knowledge base increases proportionally with the number of words and the amount of high-level NLP information. To analyze the effect of scale-up on performance, we increased the knowledge base size by adding more concept sequences and phoneme sequences. As shown in Figure 4, the execution time ranges from 10 seconds for 1.4K nodes to 31.5 seconds for a 9K node knowledge base. The total execution time increases linearly. The majority of processing time is spent in the propagation phase. Only a small portion of the code is serial and cannot be executed as SNAP array instructions.

The scalability of PASS compares favorably to a sequential implementation. Identical experiments were performed on a single processor, as shown in Figure 5. The execution time on a single processor also increased roughly linearly. However, the proportionality constant is significantly larger for the uniprocessor. Specifically, the speedup from parallelism ranged from 2.13-fold for the basic domain up to 10.7-fold for a 9K node knowledge base, with the amount of speedup increasing as the knowledge base grows.

The increase in processing time is primarily influenced by the structure of the knowledge base. First,

KB Size [nodes]	SNAP-1 [sec.]	Uniprocessor [sec.]	Speed-up
1400	10.1	21.5	2.13
1600	11	32.8	2.98
1800	11.7	37.3	3.19
2200	12.8	57.3	4.48
2400	13.5	61.6	4.56
2800	15.2	77.4	5.09
3200	16.3	98.2	6.02
3600	17.4	110.4	6.34
4000	18.7	131.7	7.04
5000	21.5	167.3	7.78
5500	22.6	191.5	8.47
6500	25.5	233.8	9.17
7500	28.1	270.5	9.63
9000	31.5	337	10.7

Figure 5: Execution Time on 16-cluster SNAP vs. Single TMS320C30 Processor

consider an efficient parallel implementation for a knowledge base which grows hierarchically. Logarithmic time performance will be provided up to the number of processors available while each increment in knowledge base size larger than this will increase execution time linearly [3]. The logarithmic time behavior is derived from the critical path length of marker-propagation which is roughly determined by the depth of the hierarchy. However, a linguistic knowledge representation typically introduces nodes at predetermined levels for the concept and phoneme sequences which are added to the knowledge base. Therefore, the depth remains relatively fixed while the breadth increases. In this case, a perfect allocation of the knowledge base can provide performance approaching constant time on a parallel machine with sufficient processing resources. For larger increases in knowledge base size, linear performance occurs. In PASS, the addition of new nodes does not significantly change the knowledge base depth. Since meaningful increments in knowledge base size exceed the number of processors in the SNAP-1 array, linear performance is obtained.

Processor Speed-up

Figure 6 shows the effect of varying the number of processors while the size of the knowledge base is held constant. Since the capacity of a single cluster is limited to 1024 nodes, a 2.4K nodes knowledge base was used with 4 or more clusters. For each cluster configuration, execution time was measured for different

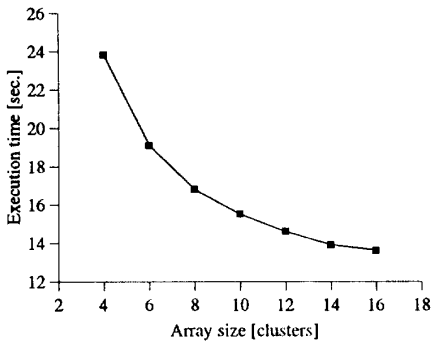

Figure 6: Processing Time vs. Number of Clusters.

sentences and the average processing time was calculated. In general, the performance improves as the number of processors is increased. The amount of improvement does level off though when a large number of clusters are used on the basic domain because the processors are no longer fully utilized. However, if a larger knowledge base is used then it is possible to take advantage of the available processing power and higher speedups can be obtained.

To further increase the speed performance we are developing a semantically-driven allocation scheme which assigns nodes from one concept sequence to the same processing element. This distributes the available parallelism evenly while reducing communication overhead associated with marker-propagation. In addition to improving the propagation phase, it is possible to optimize the non-propagation SNAP instructions and further reduce the total execution time. Additionally, the full 32 cluster configuration should provide a significant performance improvement for large knowledge bases.

4 Conclusion

We have developed an integrated speech understanding algorithm based on memory-based parsing and parallel marker-passing schemes. We have provided a parallel solution to speech-specific problems such as insertion, deletion, substitution, and word boundaries. In our approach, meaning representations constructed as a result of memory-based parsing can be applied not only to generate an output sentence but also to provide information for applications such as high-level inferencing and speech translation. The experimental results demonstrate the benefits of the parallel computational model for the integration of speech and NLP. We are now installing the Radiology domain provided by the Speech Systems Incorporated consisting of a vocabulary of approximately 2K words and a network of approximately 10K semantic nodes.

References

[1] Anikst, M.T. and Trawick, D.J., "Training Continuous Speech Linguistic Decoding Parameters as a Single-Layer Perceptron," *Proceedings of International Joint Conference on Neural Networks*, Vol. 2, 237-240, 1990.

[2] Chung, M. and Moldovan, D.I., "Memory-based Parsing on SNAP: Integrated Syntactic and Semantic Analysis," *Technical Report* PKPL 91-10, Department of Electrical Engineering-Systems, University of Southern California, 1991.

[3] Chung, S. and Moldovan, D.I., "Modeling Semantic Networks on The Connection Machine," *Journal of Parallel and Distributed Computing*, February 1993.

[4] DeMara, R. and Moldovan, D.I., "The SNAP-1 Parallel AI Prototype," *Proceedings of the 18th Annual International Symposium on Computer Architecture*, 1991.

[5] Giachin, E.P. and Rullent, C., "A Parallel Parser for Spoken Natural Language," *Proceedings of IJCAI*, 1537-1542, 1989.

[6] Huang, X. and Guthrie, L., "Parsing in Parallel," *Proceedings of COLING-86*, 140-145, 1986.

[7] Lee, K.F., "Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The Sphinx System," *Technical Report* CMU-CS-88-148, Department of Computer Science, Carnegie-Mellon University, 1988.

[8] Meisel, W.S., Fortunato, M.P. and Michalek, W.D., "A Phonetically-Based Speech Recognition System," *Speech Technology*, Apr/May 1989.

[9] Waltz, D.L. and Pollack, J.B., "Massively Parallel Parsing: A Strong Interactive Model of Natural Language Interpretation," *Cognitive Science* 9, 51-74, 1985.