N92-16595

Extensions to the Parallel Real-time Artificial Intelligence System (PRAIS) for Fault-tolerant Heterogeneous Cycle-stealing Reasoning

David Goldstein
Faculty Associate
goldstn@cse.uta.edu

University of Texas, Arlington
Automation and Robotics Research Institute
7300 Jack Newell Blvd S.
Ft Worth, Texas 76118
USA

Abstract. Extensions to an architecture for real-time, distributed (parallel) knowledge-based systems called the Parallel Real-time Artificial Intelligence System (PRAIS) are discussed. PRAIS strives for transparently parallelizing production (rule-based) systems, even under real-time constraints. PRAIS accomplished these goals (presented at the first annual CLIPS conference) by incorporating a dynamic task scheduler, operating system extensions for fact handling, and message-passing among multiple copies of CLIPS executing on a virtual blackboard. This distributed knowledge-based system tool uses the portability of CLIPS and common message-passing protocols to operate over a heterogeneous network of processors. Results using the original PRAIS architecture over a network of Sun 3's, Sun 4's and VAX's are presented. Mechanisms using the producer-consumer model to extend the architecture for fault-tolerance and distributed truth maintenance initiation are also discussed. Also, recently designed approaches and extensions, including improvements to RETE and an entirely new pattern matching algorithm to meet hard-real-time deadlines are discussed.

This paper is deliberately presented at a high-level, discussing the real-time, fault-tolerance, and distributed nature of the architecture as more detailed descriptions of the work are available elsewhere [Gol90][GT91][Gol91].

0.0 Introduction

Real time artificial intelligence (AI) is an ideal application for parallel processing. Many problems including those in vision, natural language understanding, and multi-sensor fusion entail numerically and symbolically manipulating huge amounts of sensor data. Real time reasoning in these domains is often accomplished via specialized computing resources which are often (1) very difficult to use, (2) very costly to purchase (as in the $250,000 - $2,000,000 PIM [GL]), and (3) guarantee only fast- not real time - performance.

This paper extends some of the ideas behind PRAIS, the Parallel Real-time Artificial Intelligence System, a cost-effective approach real-time computing combining the 'C' Language Integrated Production System, TCP/IP and some novel concepts in pattern matching and real-time control to provide a flexible development environment for distributed knowledge-based systems. The goals of the system are to simplify parallelization, increase portability, and maintain a consistent knowledge representation throughout the system. The system accomplishes these goals by providing transparent scalability of fielded CLIPS applications and by cycle-stealing small amounts of resources over large networks of existing processors.

1.0 Original Blackboard Architectures

The blackboard architecture [Nii86] has probably been the most successful architecture for addressing complex problems where control structures were not well-defined. KBS's using this architecture feature multiple, independent knowledge sources (KS) each of which reasons about a portion of the domain. Knowledge sources share a global data structure (the metaphorical "blackboard") to share information, in an analogy to experts examining data and hypothesizing solutions on an actual blackboard.

Parallel versions of blackboards provide several advantages. First, each knowledge source may have its own knowledge-base (KB - a database of knowledge driving reasoning), thereby partitioning the system's knowledge and reducing rule interactions. This simplification generally making the system easier to understand and more predictable. Blackboards also facilitate intuitive, hierarchical problem-solving; results from lower level knowledge sources can be used to drive the reasoning processes of higher level knowledge sources. The hierarchical development of hypotheses is very useful, especially useful for problems where disparate data is encountered from multiple sources (e.g. vision, multi-sensor fusion).

An illustration of a real time blackboard system for music generation is depicted in Figure 1. At any given time the system might receive a variety of auditory inputs. These inputs are examined by signal processing resources to extract and place on the blackboard primitives such as frequencies, pulse widths and pulse intervals. These primitives are then used by other processors to determine notes, "instruments", pauses, and durations, which are in turn combined to ascertain tempos, progressions, chords. At the highest levels of processing these deductions are combined with music styles, artistic profiles, scores and music theory to predict future sensor inputs and generate appropriate auditory output.

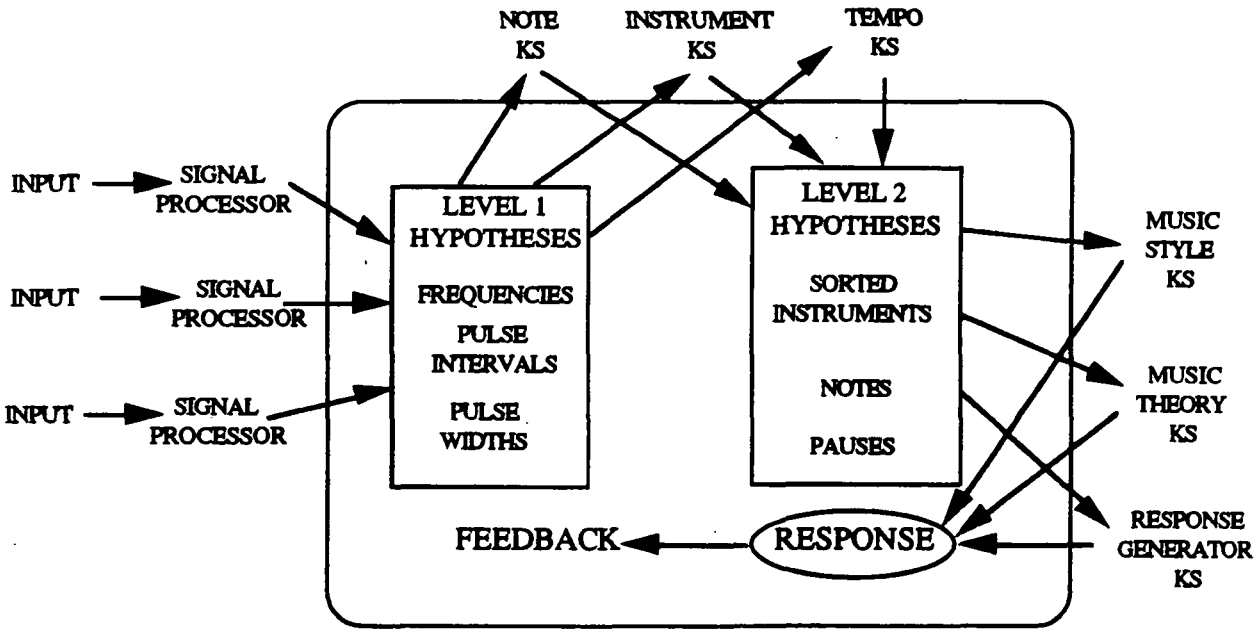

Figure 1. Hierarchical Blackboard Processing

2.0 Directed Blackboards

Directed blackboards is an architecture explicitly designed for forward-chaining production systems to accommodate real-time processing over a heterogeneous network of processors derived from examining numerous investigations into blackboard processing. Like many architectures

derived from blackboards, directed blackboards attempt to improve the basic blackboard model by improving the control mechanisms, reducing the event scheduling required, and minimizing the amount of information that must be distributed (via the blackboard).

3.0 Message Flow and Control

Message-passing among knowledge sources in this architecture is facilitated by associating a goal to each knowledge source. Each goal is managed by a Dijkstra-like guard which administers the actual communications of messages via a single-entry, single-exit point for all information. Messages flow from producing knowledge-sources (as per the producer-consumer model), to the information guards, to consuming knowledge-sources. This model assumes that each information guard knows the recipients of its information; this assumption is fulfilled by analysis of an application's productions to associate knowledge-sources which might use information (which satisfies a given patterns to goals represented by the pattern), see Figure 2: Message Traffic. Further, the information guards administer fault-tolerance algorithms and initiate distributed truth-maintenance algorithms for fault-recovery, essentially isolating these aspects of the knowledge-based system in the communications functions. Finally, although control resides almost solely in the information guards, each processor in the system shares in the distributed control of not only message traffic, but also information focus, in some fashion.

The information guards presented here can act as backing stores of the data transmitted-without the overhead of reasoning upon facts as performed in true backing processors - while the goals are currently being processed. Further, once goals are no longer being currently processed, transaction management can be performed to save the state of the system prior to operating upon new goals. This procedure treats the state-saving algorithms associated to forward-chaining as a nested transaction, with each goal change as a child transaction.

Heterogeneous networks of processors are easily accommodated in this architecture by using standard communications protocols over internet (TCP/IP). In keeping with the philosophy of the underlying inference engine, the 'C' Language Integrated Production System (CLIPS), many of the communications parameters - such as message packet size, packet destinations, etc. - are stored as facts and can be inferenced upon. Because communications functions are isolated in the information guards, evolving communications standards can easily be accommodated. The information contained in and the resources (such as processors) used by real-world systems can be easily accommodated and interfaced with via the same communications mechanisms; the information guards care not whether the producer or consumer of information is a knowledge-based system or a military simulation, as long as communication proceeds using an established protocol.

4.0 Concurrent Processing

The use of network resources to facilitate concurrent processing is straightforward at a coarse level of granularity; placing individual knowledge sources on their own processors in a multi-processing environment is intuitive and has been incorporated in many systems. The next step in increasing concurrency requires partitioning individual knowledge sources. Many message-based systems strive for "rule-level" parallelism, but such parallelism can be trivially accomplished via placing one rule in each knowledge source and running one or more processes containing knowledge sources on each processors. Several other types of parallel processing that are currently being explored include "greedy processing" where multiple rule firings occur internal to a processor before dispersing the information to other processors, based upon a mathematical estimate of the usefulness of the information to other processors, "rule-level" parallelism - but with slept threads instead of processes, and MIMD transputer matching algorithms. Several new match algorithms have already been internally developed for handling real-time processing and to better the performance of RETE, and the parallelization of these algorithms is under investigation.

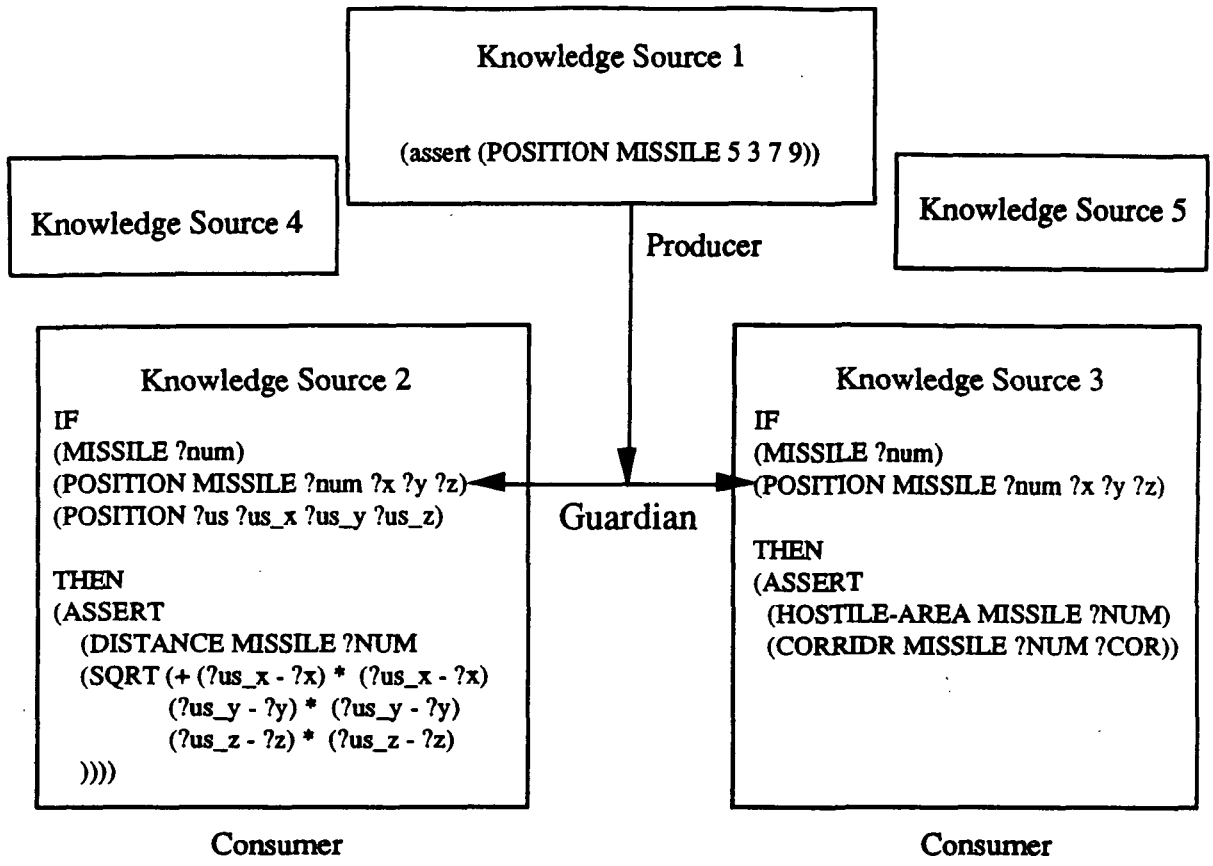

Figure 2. Message Traffic

5.0 Real Time Control Mechanisms

Our original approach for real-time control features dynamically prioritizing tasks based upon criticality and timeliness; each task has a time-varying salience function which accounts for how critical its decision is to the overall system in its current state. The salience of a task is initially low, and increases as the task becomes more important until it become mandatory. Untimely tasks or those of lesser importance can be dropped by the system. Tasks are scheduled for execution based upon a hypothesis of the task's usefulness and likelihood to complete. This algorithm employs resource estimates of the task derived via directed acyclic graphs generated during preprocessing and user guidelines as to the task's importance for generating and using the time-varying salience function (see Figure 3). This approach is in stark contrast to the typical - and computationally expensive - approach of scheduling tasks such that their hard-real-time constraints (deadlines) are met via meta-reasoning. The author feels that planning is a very expensive process, especially for large numbers of tasks, by far exceeds the time frames of executing "ladder-logic" that many real-time applications actually use (such as robotic control). Since the publication of this technique, other parties interested in real-time control, such as Boeing, have investigated similar measures [EB91].

Finally, an "anytime algorithm" extension to RETE is currently under investigation. Such an algorithm should ideally provide an answer from a knowledge-based system regardless of how little time the system is given for reasoning, with the accuracy of the answer proportional to the time allotted for reasoning. Combined with the interruptable reasoning features that have already been placed in PRAIS, this should permit a system to take the best course of action at any time, regardless of system demands.

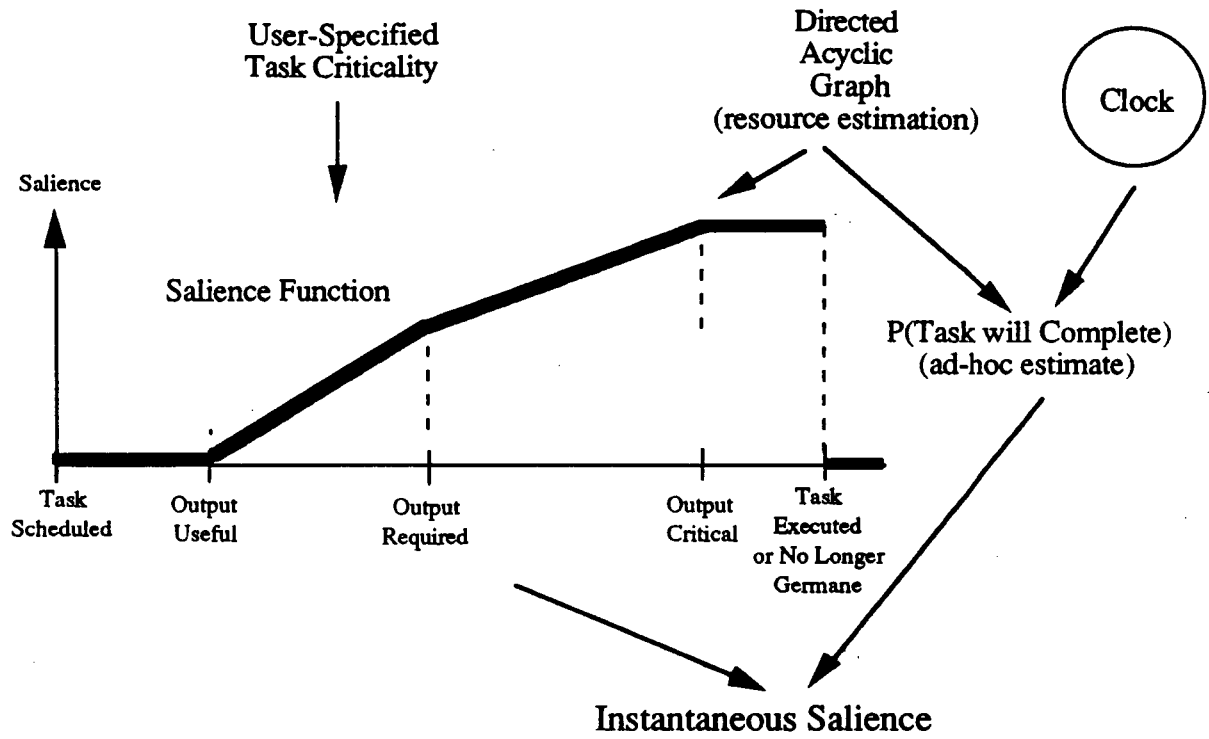

Figure 3. Time-Varying Salience for Task Scheduling

6.0 Knowledge Representation

The knowledge representation used and reasoning processes permitted are virtually identical to those already used by CLIPS. The rule format is depicted in Figure 4: Production Format. Assertions and retractions are handled exactly as there are normally in any serial version of CLIPS, but if the information change should have some affect on some global goal, the goal must be specified.

```
(rule {rule-name}
            (salience {priorities})
            (importance {mandatory/optional/dropable})
            (goal {goal_name})
      ({pattern 1 to be matched as a tuple})
      •
      • ({left-hand-side patterns})
      •
      ({pattern n to be matched as a tuple})
=>
      ({action 1})
      •
      •
      •
      ({right-hand-side action n})
)
```

Figure 4. Production Format

7.0 An Example Application

Presented below is a rule (Figure 5: Rules) pertaining to an image processing application. The $WHERE fact determines where future messages (facts) are to be passed. For fairness, the application partitions the work according to tasks to be performed, and not the data; many experimenters use image processing to demonstrate concurrent processing because image processing parallelizes very easily (by assigning equal size rectangles in the image to each processor). Each processor in this experiment must operate over all the image data, and pass its results to some other processor for further processing.

```
(defrule connect4 "determines if a point is 4-connected"
        (pt ?x ?y)
        (pt ?x =(- ?y 1))
        (pt ?x =(+ ?y 1))
        (pt =(+ ?x 1) ?y)
        (pt =(- ?x 1) ?y)
=>
        (assert        ($WHERE csr)
                       (connected4  ?x  ?y))
)
```

Figure 5. Rules

Performance characteristics (with respect to speedup) vary widely from one application to another (because of complexity of the RETE net, size of the factbase, number of facts sent per message, etc) and network resources employed (such as processor types, operating systems and network transmission media). Each of these drives a variety of underlying computational concerns; network and factbase sizes can cause disk swapping while operating systems and processor types can require different conversion strategies at the byte (bit) level.

The current implementation requires approximately one and one-half times as long to receive a transmitted fact via internet as it takes to deduce the fact itself and over seven times the time to send the same fact via internet. Therefore, if decidedly different tasks are worked upon concurrently - perhaps using "island driving" techniques - concurrent processing could yield almost linear speedup. However, with small rule networks (as experimented with here) and geographically separated resources (twenty miles apart) the speedup was not nearly linear.

8.0 Status

The Parallel Real-time Artificial Intelligence System (PRAIS) has been implemented to provide coarse-grained parallel processing over a heterogeneous network of machines, including Sun 3's, Sun 4's, Transputers and DEC VAX's. At the time of this writing the knowledge-based system shell has been modified to accommodate real time processing. The communications algorithms (with an accompanying partially ordered indexing system) have already been implemented. A large number of operating systems issues are currently being addressed, as well as fault-tolerance algorithms and the aforementioned pattern-matching algorithms. Past, current and anticipated PRAIS application areas include real-time sensor-fusion, distributed simulations, robotic control at the workstation level, and parallel planning. Larger rulebases and fault-tolerance/distributed

control experiments will be presented in future papers as more and larger research projects use the system.

9.0 Conclusions

PRAIS already offers a variety of advantages such as:

- heterogeneous hardware capability,
- uniform data flow through the system,
- real time control via dynamic scheduling,
- data - as opposed to algorithm - driven requirements, and
- the use of standard programming practices.

Future capabilities to be incorporated include fault-tolerance, automatically scalable applications, and distributed truth maintenance. This system strives to permit serial code to be converted into a parallel, real time KBS by incorporating many desirable features and functions at low levels of processing.

10.0 References

[CG88] C. Culbert and J. Giarrantano, CLIPS Reference Manual Version 4.2, Artificial Intelligence Section Lyndon B. Johnson Space Center, Houston, Texas, April 1988.

[EB91] W. Erickson and L. Baum, "Real-Time Erasmus", in Proc. of Blackboard Systems Workshop Notes from the Ninth National Conference on Artificial Intelligence, American Association for Artificial Intelligence, Anneheim, CA, July, 1991.

[Gol90] D. Goldstein, "PRAIS: Parallel Real-time Artificial Intelligence System", Fourth International Parallel Processing Symposium Proceedings, Volume 3, Fullerton, CA, USA, 1990.

[Gol91] D. Goldstein, "Integrating Knowledge-bases into Heterogeneous Networks of Processors for Real-World, Real-time Systems", in Proc. of the International Joint Conference on Artificial Intelligence's Integrating Knowledge-bases into Real-world Systems Workshop, Sydney, Australia, USA, 1990.

[GT91] D. Goldstein and J. Tiernan, "Heterogeneous Distributed Knowledge-based Systems", in Proc. of the American Association for Artificial Intelligence 1991 Conference's Workshop on Heterogeneous Systems, Anneheim, California, July, 1991.

[Nii86] P. Nii, "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures", The AI Magazine, Summer 1986, pp. 38 - 53.