A PARALLEL BLACKBOARD SYSTEM FOR ROBOT CONTROL

Hugo Velthuijsen
Ben J. Lippolt
Jeanette C. Vonk
Netherlands PTT, Dr. Neher Laboratories
P.O. Box 421, 2260 AK Leidschendam, The Netherlands

ABSTRACT

The blackboard architecture is a powerful expert systems architecture. Its main advantages are flexibility of control and integration of different kinds of knowledge representations and inferencing techniques. At our Laboratories we built a blackboard shell, based on the principles as developed in the system Hearsay-II and the blackboard shells Hearsay-III and BB1. This paper describes how we modify this blackboard shell, in order to use it for the control of a robot cell. This cell consists of a number of independent devices, which will have to execute their tasks concurrently. This means a modification of the blackboard shell towards parallelism. Ve see great possibilities for using our system not only for simulating robot cell configurations and prototyping flexible control of robot cells, but also for other systems with multiple processors.

I INTRODUCTION

This paper treats an engineering topic which involves two different subfields of Artificial Intelligence (expert systems and robotics). Ve chose to use an expert system with a blackboard architecture for the control of a robot cell.

Before we describe this application we will give an introduction to robot cells and the blackboard architecture.

A. Robot Cells

A robot seldom executes its tasks independently of its environment. Usually the environment of a robot consists of a number of different devices which cooperate to perform a given task. These devices could include, among others, visual and tactile sensors, separate processors for executing computational tasks, operators which import objects into or export them out of the environment, and the actual robot arm and gripper. The environment of a robot even could include other robots. The robot together with all the devices which cooperate in the environment to perform the given task are called a robot cell. It is obvious that a robot cell like this requires a supervisor to control the correct interaction of all the components of the robot cell. For a description of a robot cell see e.g. (Kak, Boyer, Chen, Safranek and Yang 1986).

At this moment research in the area of robotics forms a major topic at our Laboratories of the Netherlands PTT. Part of the research is done by developing a robot cell to test the theoretical findings in a practical setting. This cell needs a supervisor with the following features:

1. The supervisor has to be able to deal with multiple components with different functionalities and different representations of these functionalities.
2. The supervisor has to be able to determine which component should perform which task at a certain moment, according to the current situation of the robot cell and the task at hand. This also includes the detection and handling of errors.
3. The way the supervisor reacts to certain situations in the robot cell must be easily modifiable in order to experiment with different methods of executing certain tasks.
4. It must be possible to add, remove, and exchange components without changing the supervisor.
5. It is necessary that multiple components can be executing concurrently.

B. Blackboard architecture

Of the five features, which were mentioned above and which are needed for the supervisor of our robot cell, the first four are provided by the blackboard architecture. We built a blackboard shell (Lippolt, Velthuijsen and Vonk 1986) with these features, using articles about the system BB1 as our main guideline. The matter of parallelism remained to be examined. An attempt to introduce parallelism to the Hearsay-II system is described in (Fennell and Lesser 1977). (Ensor and Gabbe 1985) also describes how the blackboard can be modified to support parallelism. As these articles limit themselves to discussing modifications of the blackboard, we intend to focus on the requirements for the control unit, although the modifications of the blackboard will also be included.

For information about the blackboard architecture we can recommend the following articles: (Erman, Hayes-Roth, Lesser and Reddy 1980) on Hearsay-II, (Erman, London and Fickas 1981) on Hearsay-III, and (Hayes-Roth 1985) on BB1.

II PROPOSED MODIFICATIONS

When implementing a blackboard system with sequential execution of knowledge sources (KSs), the following implementation of a succession of actions can be used, as we did in our initial blackboard shell.

During the execution of a KS, every change of the blackboard results in the creation of a special formatted data object, called an event, which records the change. After the execution of a KS, the control unit checks for all KSs whether they are interested in the occurrence of one of these events, or any combination of events. If there is a match, another data object is created which records the information significant to this KS and this particular combination of events. In the literature this is called a knowledge source activation record (KSAR). At this stage this KSAR is called triggered. Next for all triggered KSARs the precondition is checked which is mentioned in their respective KS descriptors. This is a procedure which can be used to test whether this KSAR really is applicable in the current situation of the problem solving process. This provides a more sophisticated way of checking the applicability of a KS then when one would be restricted to using triggers. Also this precondition can be used to actualise the values of the parameters which are needed by the KS. If the test has a confirming result, the KSAR becomes what we will call invocable. Then for all invocable KSARs their priority is determined and the one with the highest priority is selected and executed. This ends the control cycle.

Our blackboard shell was built according to this description. As its main features it further supports the use of a blackboard for control purposes and an unlimited number of blackboards for domain knowledge (this is done to facilitate the structuring of the domain knowledge) and the use of multiple knowledge representation methods for the action parts of the KSs (these include the languages LISP, PROLOG, and POP-11 and it is also possible to make use of rule based, frame based, and object oriented inferencing techniques). A blackboard is subdivided into blackboard levels and the levels are subdivided into units which are sets of attribute value pairs. In order to help the knowledge engineer in the task of building a system, our blackboard system is equipped with a wide variety of (optional) trace facilities. At one moment during the control cycle it is possible to examine all relevant data, such as the contents of the blackboard(s), the events which occurred during the latest execution of a KS, the triggered and invocable KSARs, and the chosen KSAR.

An architecture as described here works fine if all KSs are to be executed sequentially. But it is evident that it is not sufficient if multiple processors are available for concurrent execution of KSs. we can distinguish three main aspects where this architecture has been modified to accommodate parallel KS execution.

First, the control unit has to be able to select a suitable KS for execution any time a processor gives a signal that it has finished its last task. In the architecture as described earlier, this is not possible because the control unit is not an available process when a KS is being executed. Second, the blackboard handler has to be able to service requests for reading or writing on the blackboard which arrive at the same time. Moreover, the requests of one KS can interfere with the requests of another KS and, if this is not handled properly, the consistency of the data on the blackboard can be violated when independent KSs work on the same information on the blackboard. Third, an environment must be created which makes it possible to execute multiple processes concurrently. We will describe these aspects in more detail.

A. A Monitoring Scheduling Mechanism

The main modification of the control unit, necessary for parallel KS execution, is to delegate the execution of the KSs from the control unit to the processors.

If a processor has terminated a task, it sends a message to the control unit that it is free to accept another task. For all processors these messages arrive in a single list. The control unit updates the sets of triggered and invocable KSARs, using the events, which occurred since the last updating took place, and the current state of the blackboard. Next the control unit checks which invocable KSARs can be executed by the free processors. For these KSARs a priority is computed, the most promising is selected for execution, and a command is sent to an applicable processor to execute the chosen KSAR. After this command, the control unit does not wait for the termination of the KSAR execution, but immediately continues with the selection of a task for the next idle processor. If no invocable KSAR was found for these processors, the original message is put back in the list.

One can imagine a situation where KSs can be executed on different processors, but that there is, possibly, a preference for one processor depending on the necessary processing capacity. Our system is equipped for this.

B. The Blackboard Handler

Although the matter of modifying the blackboard handler in order to be able to cope with concurrent access to data on the blackboard has been treated extensively in (Fennell and Lesser 1977) and (Ensor and Gabbe 1985), we think it is useful to summarise the various possible adjustments.

We implemented the blackboard handler as a process with a list (in order to handle different priorities) of requests for reading, writing, or modifying units on the blackboard. This way the incoming requests can be sequentialised. For maintaining the consistency of the data on the blackboard a few suggestions are found in the literature.

1. A KS could ask the blackboard handler to

forbid any other KS to read or write those units which that KS is possibly going to use. Such locks could be restricted to only one unit, but could as well be extended over entire levels and even blackboards.

2. A KS could add a tag to units which it is going to use. Whenever other KSs make changes to a tagged unit a message is sent to the KS which placed the tag.
3. Vhen a KS wants to make changes to the blackboard the KS could check (after the relevant sections are locked for reading) whether the information on the blackboard coincides with the assumptions about the contents of the blackboard, made by this KS. If the assumptions are still valid, a request for a write lock is made for those sections of the blackboard which are to be changed and then the blackboard can be changed. If the assumptions are no longer valid, the attempt to change the blackboard is cancelled.

The first solution has one main disadvantage. Often it is not possible to foretell exactly which blackboard units are going to be operated on. This means that in some cases the locks could become very extensive, thus blocking the execution of other KSs. A disadvantage of the second option is the high cost in overhead. It seems that the third option is most promising. Option 3 needs the facilities as mentioned under 1, but in this case the locks are installed for a shorter period. Also, in this case, the locks often don't need to be as extensive as under option 1.

It should be mentioned that in a lot of applications, as in ours, the interference between different KSs can be kept low by only selecting KSs for concurrent execution which operate on disjoint sections of the blackboard. In order to be able to use option 3 we implemented the first option. The knowledge engineer is recommended only to use option 3.

C. Concurrent Process Execution

To enable the blackboard system to execute KSs concurrently, we divided the system over various processes. This means one process for the control unit, one process for the blackboard handler, and one process for each of the processors which are to be represented. For simulation we used a VAX under the operating system UNIX* and execute the processes in timesharing. During the simulation stage the action parts of the KSs often only consist of a procedure which modifies the blackboard in a fixed way after a certain amount of time. This means that using timesharing doesn't create a real limitation. Vhen the system is being used for the actual control of a robot cell a similar set up will be chosen. Then on the controlling processor, the processes for executing the KSs will consist of gateways to the processors where the action parts of the KSs are actually executed.

*URIX is a trademark of AT&T Bell Laboratories.

III CONCLUSIONS

We described how we modified a blackboard shell (built along the descriptions of Hearsay-III and BB1, as found in the literature) in order to support parallel execution of KSs. We showed why we think that a parallel blackboard shell is a useful tool for prototyping and simulation of robot cell control systems. We implemented this shell in a UNIX environment in order to simulate the execution of concurrent processes. A next step would be to implement a control system for actual robot cell control. Although the blackboard architecture might not be efficient enough for real time robot cell control, we are convinced of its merits during the development stage of robot environments.

REFERENCES

1. Ensor, J.R. and J.D. Gabbe, "Transactional blackboards." In Proc. IJCAI-85. Los Angeles, USA, August, 1985, pp. 340-344.

2. Erman, L.D., F. Hayes-Roth, V.R. Lesser, and D.R. Reddy, "The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty." Computing Surveys 12:2 (1980) 213-253.

3. Erman, L.D., P.E. London, and S.F. Fickas, "The design and an example use of Hearsay-III." In Proc. IJCAI-81. Vancouver, Canada, August, 1980, pp. 409-415.

4. Fennell, R.D. and V.R. Lesser, "Parallelism in Artificial Intelligence problem solving: a case study of Hearsay-II." IEEE Transactions on Computers C-26:2 (1977) 98-111.

5. Hayes-Roth, B., "A blackboard architecture for control." Artificial Intelligence 26:3 (1985) 251-321.

6. Kak, A.C., K.L. Boyer, C.H. Chen, R.J. Safranek, and H.S. Yang, "A knowledge-based robotic assembly cell." IEEE Expert 1:1 (1986) 63-83.

7. Lippolt, B.J., H. Velthuijsen, and J.C. Vonk, "BLONDIE: a blackboard shell." Memorandum 1404 DNL/86, Netherlands PTT, Dr. Neher Laboratories, Leidschendam, The Netherlands, November 1986.