CUCS-16-81

Specialized Hardware for Production Systems

Salvatore J. Stolfo
and
David Elliot Shaw
Department of Computer Science
Columbia University
New York, New York 10027

April 1981

INTRODUCTION

One of the least explored areas which we are investigating as part of our research on Parallel Architectures and VLSI Systems is the design and analysis of machine architectures specially adapted to the highly efficient implementation of Production Systems (PS) [Davis1976, Newell1973, Rychener1976]. Several relatively effective Artificial Intelligence (AI) programs have been written ,recently demonstrating the suitability of rule-based representations for many applications requiring extensive domain expertise, for example [Davis1976] and [Buchanan1969]. As has been reported by several researchers, rule-based systems offer an appropriate formalism for acquiring information from human experts, and at the same time are easily implemented and readily modifiable and extendable. It seems only reasonable then that special purpose hardware for the efficient implementation of PSs would be of particular importance for the next generation of AI systems.

Production Systems

A production system is defined by a set of rules, or productions, which form the Production Memory (PM), together with a database of assertions, called the Working Memory (WM). Each production consists of a conjunction of pattern elements, called the left-hand side (LHS) of the rule, along with a set of actions called the right-hand side (RHS). The RHS specifies information which is to be added to (asserted) or removed from WM.

In operation, the PS repeatedly executes the following cycle of operations:

1. Match: For each rule, determine whether the LHS matches the current environment of WM.

2. Select: Choose exactly one of the matching rules according to some predefined criterion.

3. Act: Add to or delete from WM all assertions as specified in the RHS of the selected rule.

Goal of the Research

In practical applications of the sort anticipated by most researchers in the field of AI, the set of productions (and hence the set of LHS patterns against which WM must be matched on each cycle) are expected to typically be quite large. Even in the case of the larger experimental PS's (MYCIN and DENDRAL, for example), current users reportedly experience frustration based on the length of time required for operation. To fulfill their promise for the very-large-scale embodiment of domain-specific expertise, PS's having an order of magnitude more rules may well be required, making the question of efficiency a potentially critical concern.

Because the matching of each rule against WM is essentially independent of the others (at least in the absence of contention for data in WM), it is natural to attempt a decomposition of the matching portion of each cycle into a large number of tasks suitable for physically concurrent execution on parallel hardware. While this task is in fact considerably more complicated than it might first appear, we believe the immense potential value of a powerful and highly general production system machine warrants serious attention.

The immediate goal of our research is thus the formulation and critical comparison of alternative, highly concurrent architectures for the very rapid execution of rule-based AI software. Our long range goal is to physically realize such devices in hardware drawing heavily on the emerging technology of highly parallel VLSI systems.

Problem Definition

A wide variety of different PS formalisms are possible [Davis1975], each interacting significantly with design considerations for an efficient parallel architecture. For a variety of reasons, we have chosen to focus on PSs in which data elements in WM have the form of arbitrary ground literals of the first order predicate calculus, while both the LHS and RHS of each rule are conjunctions of predicates having first order terms composed of function symbols, constants and existentially quantified variables.

Some PS formalisms allow the use of universally quantified variables in the LHS of a rule. While we have investigated certain issues related to the use of universal quantification in the LHS, we expect to concentrate on the simpler case in which variables are assumed to be existentially quantified. (It should be noted that multiple instantiations of a single production are thus possible.)

Alternative Architectures

Although we believe it to be too early in our investigations to rule out any potentially feasible architectures for the construction of PS machines, our efforts to date have focused largely on three basic implementation strategies:

The first scheme is based on a large set of microprocessors, each associated with a small amount of local memory. The processors may be linked using one of several different kinds of interconnection patterns. (The particular topology chosen, for example the perfect shuffle interconnection, will have implications for both the time and hardware complexity of the resulting machine; evaluation of alternative interconnection schemes and their implications thus forms a major part of our research on this architecture.) Each production in PM is assigned to a distinct processor, along with all ground literals in WM whose predicate symbol is the same as that of one of the pattern elements of that production.

During the match phase of a given cycle, each processor compares its production with the locally resident portion of WM (which is the only part of WM that can possibly be relevant to that production), and determines whether a successful match has occurred. One of several multiple-match resolution schemes (requiring time logarithmic in the number of processors for most interconnection schemes) is used to identify a single production for execution during the current cycle. The corresponding processor then broadcasts all changes specified in the RHS of its rule to all other processors (again in a manner dependent on the exact interconnection scheme), which update their portions of WM accordingly, as necessary.

As the reader will note, this scheme requires much of WM to be replicated throughout the machine, since a number of productions may in general contain occurrences of the same predicate symbol. Furthermore, the testing of a single production forces a serial matching process within each processor. An alternative scheme we are considering is to divide each LHS into its component parts, assigning to each processor the responsibility of matching a single pattern element. A single LHS of a rule therefore is represented by a bank of communicating processors. By allowing multiple occurrences of equivalent pattern elements (relative to a renaming of variables) very little of WM will need to be duplicated. However, it is not known whether the serial matching process will be completely parallelized in this scheme since much of this process is embedded in the communication patterns between processors. The reporting of successful matches of productions against WM and the selection of a single rule for execution is the same as in the previous scheme.

The third architectural class we are considering makes use of hardware parallelism at a much lower level than the two described above. This approach, which has not yet been defined in any detail, is based on the adaptation of a distributed logic approach to associative processing. In general terms, the content-addressability of such a processor would be exploited to perform production-matching operations in parallel.

While the case for associative processing in the large-scale symbol-matching task involved in the operation of a PS is certainly a compelling one, the conventional distributed logic associative processor (e.g., see [Lee1962]) would require the following modifications to be applicable to the construction of a PS machine:

1. Adaptation to efficiently handle the case of a large number of patterns, as opposed to a single pattern matched against a large database.

2. Modification to adapt the common "broadcast" bus typical of distributed logic associative devices to the physical constraints (involving capacitive loading) of a VLSI chip.

Our preliminary analysis suggests that the high speed implementation of the project and join operations for a relational data base machine (see [Shaw1980]) may have applicability to this third architectural class.

Our work to date is primarily analytical in character, and has not involved the construction of hardware. Our later work might involve experimentation with multi-microprocessor assemblages for the construction of prototype versions of the first two architectures described above, and for the implementation in VLSI of elements of the third. We expect to develop substantial software tools both for simulation of particular architectures and as design aids for specifying hardware components.

REFERENCES

Buchanan, B.G., Sutherland, G.L., and Feigenbaum, E.A., Heuristic DENDRAL: a Program for Generating Explanatory Hypotheses in Organic Chemistry, in Machine Intelligence 4, Meltzer and Michie (eds.), Elsevier, New York, 1969.

Davis, R., Applications of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases, Ph. D. thesis, Stanford Univ., 1976.

Davis, R., and King, J., An Overview of Production Systems, Stanford Univ. AI Lab Memo, AIM-271, 1975.

Lee, C.Y., Intercommunicating Cells as a Basis for a Distributed Logic Computer, Proc. AFIPS 1962 Fall Joint Computer Conference, Spartan Books, Inc., Baltimore, Maryland, pp. 130-136, 1962.

Newell, A., Production Systems: Models of Control Structures, in Visual Information Processing, W. Chase (ed.), Academic Press, 1973.

Rychener, M., Production Systems as a Programming Language for Artificial Intelligence Research, Ph. D. thesis, Carnegie-Mellon Univ., 1976.

Shaw, D.E., A Relational Database Machine Architecture, Proc. 1980 Workshop on Computer Architecture for Non-Numeric Processing, 1980.