A Note on Another Structure Sharing Method

3Q-6

K.Hirata, H.Tanaka and T.Moto-oka

Faculty of Engineering, University of Tokyo

1. Introduction

Some machine architectures have been developed so far, on which logic programming language can be executed. Also the more unification mechanisms and internal data representation methods have been proposed. Especially we concern a fast execution mechanism for logic programming language on MIMD machine, and developed an efficient execution model suitable for the highly parallel processing.

In this paper, we will introduce yet another structure sharing method on MIMD-type inference machine and explain its concepts.

2. Unification and Reduction in Parallel Machine

For the later discussions, in this section, we will simply review the unification algorithm. A successful unification between a goal and a definition produces the intermediate results in general. The results consist of some goal clauses and their environments of bindings. Environment is the substitutions of terms for variables created by unification.

For distributed processing in MIMD machine, as processing element (PE for short) transfers goals to the other PEs frequently, it is better that PE makes a complete and compact goal. So such a form for goal is desirable. To achieve these requirements, reduction what we call must be performed. In fact, reduction is to copy the reachable data structures and to dereference until getting real values for variables.

3. Concept of Structure Sharing/Copying

The first choice point for representation of goal is whether the structures should be shared or copied. The basic idea of structure sharing is to push the tuple of the pointers for every unification onto the stack. One of the pointers refers to environment and another to the structure in definition clause (called skeleton), i.e. program. Hence the referred structure in definition clause is shared, so that when constituting new goal, the redundancy in the representation of different instances with the same skeleton can be avoided. On the other hand, structure copy is to copy the necessary skeleton into private field, and to substitute the actual values for variables whenever unification completes. So the newly created structures are not shared at that time. Up to now a lot of methods have been considered, corresponding to what level the structure is shared (copied) to.

4. Concept of Environment Sharing/Copying

The second choice point is whether the environment of the parent goal should be shared or copied when it will be commonly inherited by son goals (resolvents). This concept is important when we consider that more than one son goals are processed in parallel. In environment sharing, it is sufficient that only new informations are added, while if son goal needs the actual value for variable, dereference will take place in all of the parent environments. In environment copying, to make new whole environments for each son goal, copying operation is required, then only the necessary environment for each is copied, and practically it can make independency and locality among goals enhanced.

5. Parallelism

The parallelisms which logic programming language involves are classified into AND/OR-parallelism. OR-parallelism is easier to implement generally. In conventional implementations of sequential prolog, the following two points have been mainly considered :

(1) an efficient structure for the resolution mechanism;
(2) a space saving structure for the memory.

Moreover, to get and realize high performance and flexible control strategies in highly parallel machine, the third point is required :

(3) independency and locality of the run-time structures,

this is the most significant.

The sharing mechanism saves memory space and copying overhead, while it makes the unification algorithm and data structure complicated. In addition, it decreases independency and locality of goals drastically. Accordingly in the processing concepts which have been proposed for inference machines so far, the various compromise of copying and sharing were chosen.

6. Yet Another Structure Sharing Method

In our method, the internal representation of goal is shown in Fig.1, in which vari-sized cells are used.

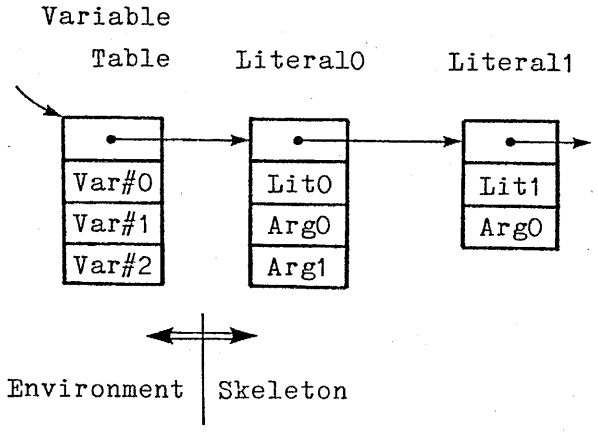

Fig.1 Internal Representation of Goal

The first linked cell contains variable bindings and the pointer to the whole skeleton of a goal. This part is called variable table cell (VT for short), which plays a role of environment. Indexing with variable number, the substitutions for variables are stored into VT with the same size as variables involved in the skeleton. In the case of initial goal, therefore, all of the binding informations are undefined. The rest part of goal is the goal's skeleton, which includes some variables and structure as a frame. If a variable exists within a skeleton and some instance is found in the corresponding field of VT, then the variable is regarded as the instance (may be the other variable).

We will show how unification and reduction are performed (Fig.2). In this example, the data necessary for the unification is placed in PE and the rest in structure memory (SM for short). Fig.2(a) depicts initial goal formation. After the unification, VT grows by the number of newly introduced variables, which are contained in the definition clauses (Fig.2(b)). If binding information is structure data, it should be stored into SM. When a data in SM is required for later unification, an operation to get it should be done. Moreover, the reduction is carried out to eliminate the unnecessary part of the goal (Fig.2(c)). In this example the reduction takes place in SM, and goal becomes compact. Practically, this operation should be done without a side effect employing copying and it is not necessary whenever unifying. As we see above, the VT and the skeleton are dynamically created and reduced.

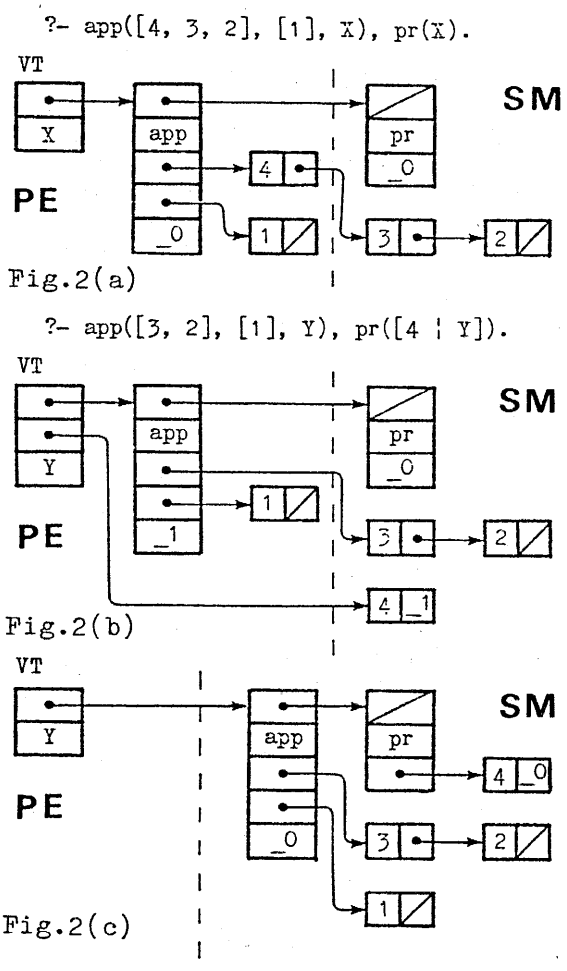

Fig.2 The Example of Unification and Reduction

7. Consideration

In our method, as every goal has its VT, goals are logically independent each other, but physically may be not. The reduction can make goals physically independent, too. An implementation of AND parallelism may require some techniques. We can implement OR parallel processing on MIMD machine efficiently.

As it is sufficient that the top parts (VT and the 1st literal) are transferred among PEs for unification, the transferred data size through the network must be small enough. To issue the reduction command to SM frequently may be able to keep the size of the top part of goal small. It is not necessary for only PEs to bear the copying overhead. However, the locality is decreased and the load of SM is increased a little, because the skeleton of goal is spread over the SMs.

8. Conclusion

We proposed yet another structure sharing method which is suitable for MIMD machine. We are now constructing the simulator adopting this method. The estimation of its performance is one of future works.

Reference

[1] Hirata,K., Tanaka,H. and Moto-oka,T., "The Architecture of The Structure Memory of PIE", 29th National Conference of IPSJ (In Japanese).