The Intelligent Channel: A Scheme For Result Sharing in Logic Programs

Simon Kaslf and Jack MInker
University of Maryland

1. Introduction

The separation of logic and control In logic programs allows the programmer to write programs whose execution Is determined by the Interpreter. This characteristic of logic programs spurred research towards diversifying the means for controlling the execution of logic programs, and towards understanding and exploring the value of parallelism In logic programming. Much of these efforts belong to the study of AND/OR-parallellsm, I.e., parallel execution of conjunctive/disjunctive goals respectively.

A brute force approach to AND/OR-parallellsm, I.e., executing every possible subgoal In a conjunctive goal has two major drawbacks: combinatorial explosion of processes and the need for many active binding environments. The latter arises due to the Interaction of shared variables In conjunctive goals.

We propose a scheme to alleviate the above difficulties. For simplicity, we demonstrate our approach with examples where atoms share a single variable. The approach Is applicable to Horn-clause logic programs [1].

2. A Model for AND/OR Parallelism

We first Introduce a model that serves as a unifying basis for constructing various schemes for parallel execution of logic programs, (see [1] for details.

OR-nodes are created when a conjunctive goal of n atoms (n>l) Is split Into n subgoals, that may be executed In parallel. When such *OR-nodes* are created as descendants of an AND-node, a new environment Is created at each of the newly created sons. The environment of an OR-node Is a set of variable templates v_i , where each variable template Is a pair:

$$v_i \quad : < in_i , out_i >,$$

v_t Is a new private variable Id, in_i, Is the location of the current Instantiation of the variable v_i, and out_i Is the location where the partial bindings for v_t are stored.

The environment of AND-node which Is the direct ancestor of the new OR-nodes Is modified with an equation:

$$x_i = UNIF(v_1, v_2, \ldots, v_n)$$

where v_i-s are all the newly created variables (Fig. l.a). The *functional* semantics of UNIF Is that of determining If the bindings for the v_i-s unify. The communication among conjunctive nodes Is determined using the *procedural* specification of UNIF.

AND-nodes are created as a result of a successful unification of a goal represented as an OR-node and some procedure head in the program.

Analogously to OR-nodes, the environment of AND-nodes is a set of variable templates. Let v be some variable that occurred in an OR-node a, and let t_1 . . . , t_n be the partial bindings for v and α obtained by creating all possible sons of a. We add the equation

$$v = MERGE(t_1, \ldots, t_n)$$

to the environment of α (Fig. l.b). MERGE performs a set union of the bindings to the variable v. As before In the case of UNIF, the operational specification of procedure MERGE Is left unspecified. MERGE-UNIF equations allow to separate the process of resolution/reduction from the process of obtaining the bindings for shared variables. This separation of process control (literal sequencing) and communication control (management of bindings) allows the Interpreter to define various communication strategies among conjunctive goals [l]. In particular, the communication structure may be determined by the setting of the variable templates In the environments of an AND/OR tree, and enforcing lazy/eager evaluation of the equations.

3. Intelligent Channel

Let P(x),Q(x) be an AND-node, and let P(x) and Q(x) be Its two OR-sons. If 't' Is a binding for P(x),Q(x) then there exists at least two paths I(P,P') and I(Q,Q') such that I(P,P') and I(Q,Q') are V-compatlble, i.e., the composed substitution along each path is unlflable with 't' for x.

Thus, It Is possible to pursue compatible computations of P and Q In parallel until they both reach the first OR-node where a decision must be made as to which path to pursue next. At the conflict point a binding Is performed at each of the parallel processes to Its own private copy of the shared AND-varlable x. This binding is reported to a channel process assigned to a shared variable. All newly created nodes are suspended until a decision is made as to which partial OR-binding to pursue. The selection criterion is based on the compatibility of *all* parallel OR-computations that originated from the conjunctive goals with the shared variable. We refer to partial bindings so obtained as *candidate bindings.* Once a candidate binding is chosen, only those suspended nodes whose execution is consistent with this binding are fired. When some partial binding is established to be a failure binding, this Information is propagated up, and all the computations contingent on this partial binding are terminated.

For example, consider the logic program below for the goal $\leftarrow P(x),Q(x)$.

$$P(x)\leftarrow P1(x).\ Q(a)\leftarrow Q1(a).\ P1(b).$$
$$P(a)\leftarrow P2(a).\ Q(c)\leftarrow Q2(c).\ Q3(b).$$
$$P(b)\leftarrow P3(b).\ Q(c)\leftarrow Q3(c).$$

A parallel execution of the program is given in Fig. 1. In Fig. l.b I(P,P1) Is compatible with I(Q,Q1), I(Q,Q2) and I(Q,Q3). The path I(P,P2) is compatible with I(Q,Q1), and I(P,P3) is compatible with I(Q,Q3) . The Interpreter selects 'a' as a candidate binding for x. In Fig. l.c 'a' was recognized to be a failure for P and subsequently for $<P,Q>$., Similarly, 'c' Is chosen for execution and Is marked as a failure binding for $<P,Q>$. As a result all the computations depending on a/x and c/x are removed yielding the snapshot In Fig. l.d. Finally, the binding *'b'/x i*s selected yielding a successful parallel search, Fig. l.e.

The evident benefits that stem from maintaining compatibility of parallel executions of OR-paths are not Immediately Implementable within the framework of existing control constructs. Below we present an algorithm, termed Intelligent Channel 1 (IC-1), that combines AND-parallelism with backtracking on partial bindings. IC-l is based on a new control construct, *SELECT BINDING* that restricts the parallel execution of multiple branches in an AND/OR tree to the branches compatible with the selected binding.

Algorithm Intelligent Channel 1 (IC-l)

To clarify the presentation of the algorithm we present a semi-synchronous algorithm In which expansions of an OR-node are obtained simultaneously.

IC-l consists of three parts: OR/AND-node execution and Channel Process execution. The creation of AND/OR nodes is as described in Section 2. OR/AND-node executions extend the creation of AND/OR-nodes with communication among conjunctive goals, and with pruning branches based on failure bindings. The Channel Process is delegated to the solution of MERGE-UNEF equations accumulated during a partial execution of a program and to the selection of candidate bindings. IC-l follows:

OR-node execution

1. If the node is null or solved report success to the parent and terminate. Otherwise:

2. If the node is enabled, expand It by generating ALL possible successors. The enabling criteria for AND/OR nodes is the selection of a candidate binding which is consistent with the composed substitution used to derive the node. The successors are generated as usual, by unifying the node with the set of axioms In the program. However, the instantiation of free variables in the node is determined by the candidate binding selected by the CHANNEL assigned to the free variable, and is obtained by accessing the "In" field of the variable template. This is how communication takes place among conjunctive goals

3. Suspend ail newly created sons until the Channel process selects a candidate binding which is consistent with the composed substitution used to generate the new sons.

4. Report ALL newly computed partial bindings to the Channel Process; I.e., all partial bindings obtained by a substitution of the form: t/x are reported to the CHANNEL x. For simplicity, in the first version we insisted on mutual exclusion in Step 4 among all active AND-node processes, I.e., only one AND-node at a time communicates Its partial bindings to the CHANNEL.

AND-node execution

1. If a node is *enabled,* create all its OR-sons, otherwise, suspend. Sons of an AND-node are created by splitting the conjunctive goal associated with the node into Independent subgoals that Interface through the UNIF equation.

2. For each new shared variable x create a shared channel data-structure denoted as CHANNEL x, and a Channel process that monitors the nondetermlnlstlc execution of its sons. This is achieved by setting the variable template of v_i to be $v_i:<in,out_i>$ for all such variables v_t that emerged from the variable x. Thus, all v_i s that emerge from x share the same current Instantiation.

3. Propagate up final/failure bindings for the node.

Channel Process Execution

1. For each CHANNEL check newly computed partial bindings against the set of candidate-bindings, thus, performing a partial solution of the MERGE-UNIF equations.

2. Whenever a failure binding is discovered, terminate the failure paths.

3. Whenever the set of candidate bindings is empty, *SELECT* a new candidate binding and activate all AND-nodes suspended on this binding.

ALGORITHM IC-l

```
Assign goal to AND-process and enable It.
Repeat until the goal is solved:
Execute all active AND-node processes.
Execute all active OR-node processes.
Execute all Channel processes.
```

4. Discussion

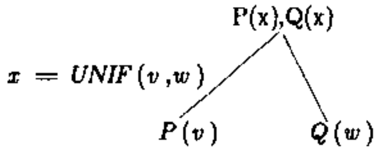
The major advantages of our approach are that only compatible computational paths are executed simultaneously, I.e., several branches of an AND/OR tree are explored In parallel while maintaining only *one active*

environment; The degree of parallelism may be monitored by modifying Step 1 in the Channel process to select multiple bindings. The combinatorial explosion that arises as a result of expanding the complete AND/OR tree is reduced, and many useless computations are eliminated.
Step 1 in the Channel Process is the key to completeness and efficiency of 1C-1. Analogous to procedure Select node in a conventional logic programming interpreter [2], select binding is the control component of 1C-1. It is not difficult to verify that if we select bindings in a breadth-first fashion, IC-1 is a complete search strategy. However, If a "depth-flrsf-llke selection strategy Is used completeness is not guaranteed.

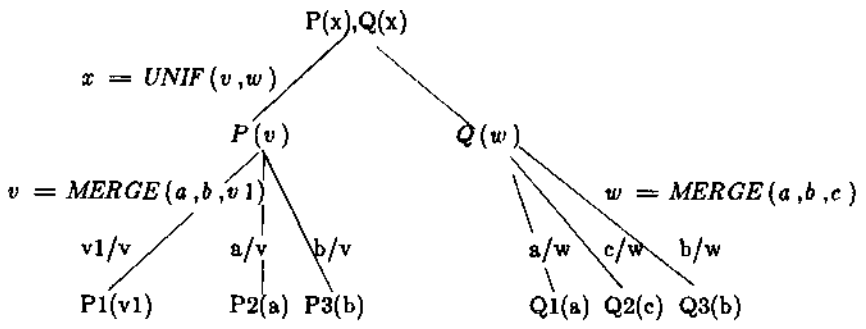
References

[1] Kasif, S., Analysis of Parallelism in Logic Programs, Ph.D. Thesis, Computer Science Dept., Unlv.of Md., 1984.

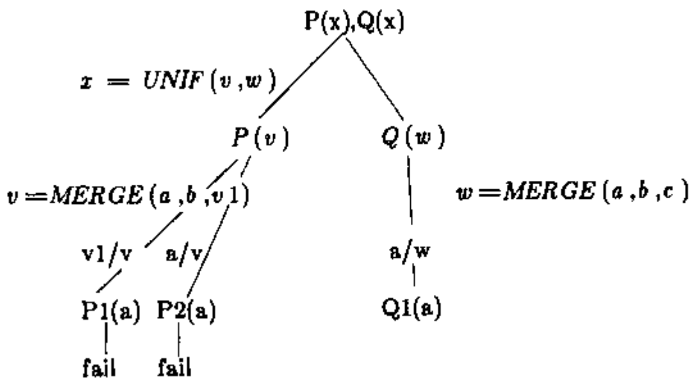
[2] Kowalskl, R. A., *Logic for Problem Solving,* North-Holland, 1979.



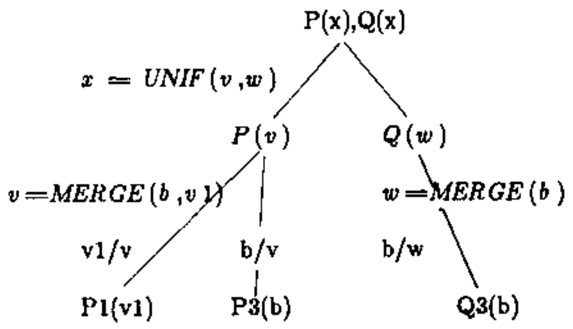
(a) OR-nodes Creation.



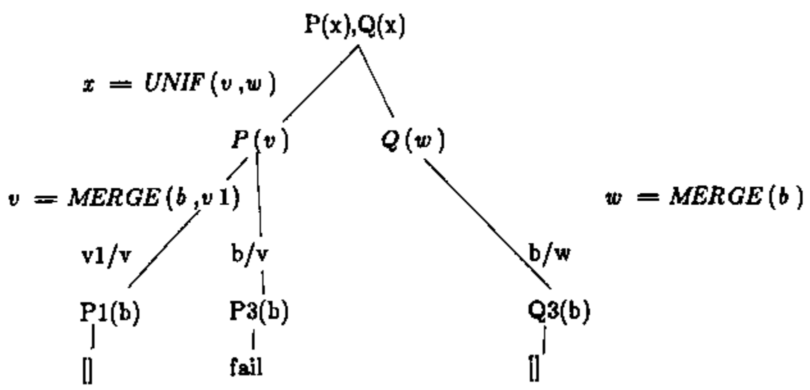
(b) AND-nodes Creation.



(c) Candidate binding = 'a'.



(d) The Tree after elimination of 'a' and 'c'.



(e) Candidate binding = 'b'

Figure 1: Intelligent Channel Execution.