

HIERARCHICAL PLANNING IN A DISTRIBUTED ENVIRONMENT

Daniel D. Corkill

Computer and Information Science
University of Massachusetts
Amherst, Massachusetts, 01003

Planning is a crucial aspect of many applications which are naturally suited to the use of distributed processing hardware. Use of a centralized planner is generally incongruous with effective distributed problem solving systems, motivating generalization of centralized planning techniques to accommodate multiple and distributed centers of planning control.

Such a generalization of Sacerdoti's NOAH (Nets of Action Hierarchies) planning system is described. This generalization involves distribution of NOAH's criticism and world model mechanisms.

1. INTRODUCTION

Planning is a crucial aspect of many applications which are naturally suited to the use of distributed processing hardware. These applications often occur in situations where sensory devices, processing capability, and devices to be controlled have wide spatial distributions and are even mobile.

Use of a centralized planner in these applications is incongruous with the development of distributed problem solving methodologies. Due to the high cost of interprocessor communication, transmitting environmental information to a centralized planner and distributing completed plans to appropriate processors is potentially expensive. The inappropriateness of centralized planners in most distributed applications motivates generalization of centralized planning techniques to accommodate multiple and distributed centers of planning control.

2. A DISTRIBUTED NOAH SYSTEM

2.1 NOAH as a Framework for Distributed Planning

Sacerdoti's NOAH [1] is a suitable candidate for generalization to a multiple center distributed planner for several reasons:

- a. Plan expansion is already localized to the expansion of individual actions. Distribution of plan expansion is simply a matter of locating plan expansion (SOUP) procedures at each processor. Separation of plan expansion from consideration of action interdependencies (criticism) allows expansion to be performed locally, prior to the necessarily non-local analysis of interactions.

- b. The integration of nonlinear and hierarchical planning techniques reduces the combinatorial growth of planning. This reduction in the size of the planning space potentially lowers the amount of interprocessor communication required in the distributed system.

- c. The generated plans retain their nonlinear representation, allowing for parallel distributed execution without additional processing to detect potential parallelism.

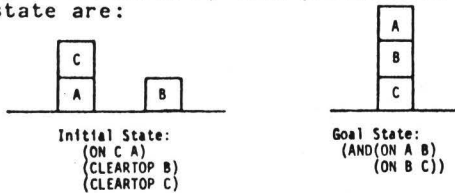
Intelligent allocation of planning task responsibility among individual processors is an important aspect of distributed planning. In the distributed NOAH system, certain high-level conjunctive subgoals are allocated to individual processors. Division between conjuncts is appropriate because conjunctive goals are initially assumed independent by NOAH. To the degree that the conjuncts are indeed independent, there is no need for interaction between processors to linearize the subplans, and the subplans can be executed in parallel. However, this approach does not guarantee a close correlation between the spatial distribution of the planning process and the spatial distribution of environmental information and of effectors required during plan execution. In the following sections, a suitable subplan to processor allocation mechanism (a Decompose Plan critic) is assumed.

To complete distribution of NOAH's planning phase, the world model and general criticism mechanisms must be distributed. The distributed critics are extensions of NOAH's original general purpose critics. The centralized versions of these critics are retained in the distributed NOAH system and are used to operate on the local plans, with interprocessor criticism performed by the distributed criticism techniques.

This research was supported by National Science Foundation Grant MCS78-04212.

2.2 Distribution of Resolve Conflicts

To show how a distributed Resolve Conflicts critic can be implemented, we look at a simple problem involving three blocks and two planning processors. Although illustrative of the interactions between distributed planning elements, blocks world problems have a high degree of interaction between actions, and the resulting plans tend to be linear in nature. The initial state of this problem (which is assumed to be known by both processors) and the goal state are:



A single processor, P1, is given the goal state, which is entered on P1's local procedural net (Fig. 1a) and expanded into a conjunction at the second hierarchical level (Fig. 1b).

P1's Decompose Plan critic allocates one of the conjuncts, (ACHIEVE(ON B C)), to a second processor, P2 (Fig. 2a). The other conjunct, (ACHIEVE(ON A B)), remains P1's responsibility (Fig. 1c). The subplan which P1 will generate to achieve (ON A B) is modelled by P2 as the MODEL node (PLAN:1a). MODEL nodes are similar to PHANTOM nodes, except that their add and delete lists represent expressions that are to be made true by actions of another processor. In this case, (PLAN:1a) contains (ON A B) on its add list. The subplan which P2 will generate to achieve (ON B C) is modelled by P1 as (PLAN:2a).

P1 expands (ACHIEVE(ON A B)) to the third level, as shown in Fig. 1d. * Since (PUT A ON B) deletes the expression (CLEAR TOP B), P1 must inform P2 of this change by sending the message (DENY(CLEAR TOP B)).

P2 performs an analogous expansion (Fig. 2b), sending (DENY(CLEAR TOP C)) to P1. **

P1 receives (DENY(CLEAR TOP C)) from P2 and enters it into the delete list of (PLAN:2a). P2 does likewise with (DENY(CLEAR TOP B)),

* For expository ease, we assume that the distributed processors do not proceed to more detailed planning levels until all messages relating to the current level have been received. Although such coordination is not required, it does simplify the planning process. Planning activity without such level synchronization is described in [2].

** Such DENY messages can be eliminated by having each processor infer the changed world state from the high level subgoal specification of the other processor. In this way, only (transparent) effects which cannot be inferred need be communicated.

entering the expression on the delete list of (PLAN:1a).

P2's Resolve Conflicts critic notices a conflict between the denied expression (CLEAR TOP B), in (PLAN:1a), and the PHANTOM node (CLEAR B). In effect, the expression (CLEAR TOP B) is a resource that will be used by both processors during plan execution. P2 will require temporary use of (CLEAR TOP B), while P1 will require permanent use (deletion). Therefore, the distributed planning system must allow P2's temporary use before granting P1's deletion. Resolve Conflicts establishes this ordering by synchronizing the distributed conjuncts to insure that (PUT B ON C) will be executed before (PUT A ON B). This synchronization is performed by inserting a signalling action, (SIGNAL:2a), into the plan (Fig. 2c) and sending (WAIT:2a(DENY(CLEAR TOP B))) to P1. When P1 receives this message it inserts a wait action, (WAIT:2a), preceding (PUT A ON B), as shown in Fig. 1e. *

During execution, (SIGNAL:2a) causes P2 to transmit a proceed message to P1. After transmitting the message, P2 continues execution. The effect of (WAIT:2a) is to suspend P1's plan execution until a proceed message is received from P2. If the message has already been received when (WAIT:2a) is executed, plan execution resumes immediately.

In addition to (SIGNAL:2a), P2's Resolve Conflicts critic inserts a second MODEL node, (PLAN:1b), into the net (Fig. 2c). This node models those actions planned by P1 which will be executed following the (WAIT:2a)/(SIGNAL:2a) synchronization. The denied expression (CLEAR TOP B) is copied from (PLAN:1a) into (PLAN:1b), indicating that it will occur after the synchronization. Temporally unrelated (incomparable) model nodes model actions to be executed by other processors in relation to actions in parallel portions of the local plan. Therefore, incomparable model nodes can overlap (even totally) in the actions they model.

P1 inserts a similar MODEL node, (PLAN:2b) (Fig. 1e), to model those actions planned by P2 which will be executed prior to the synchronization. There is an ambiguity on P1's part as to when the denial of (CLEAR TOP C) occurs in relation to the (WAIT:2a)/(SIGNAL:2a) synchronization. (PLAN:2b) represents those actions which must execute before the synchronization. (PLAN:2a) also represents those actions, as well as actions that are unsynchronized. Without additional information, P1 must leave (CLEAR TOP C) on the delete list of (PLAN:2a), because that MODEL node is the least temporally specified. The expression is tagged to indicate the ambiguity.

* In these problems, certain node names are identified by processor and an index (a, b, c, etc.). The same processor/index identification is coincidental--except with WAIT and SIGNAL actions which are paired using this identification.

P1 finds no additional criticism to perform at the third level, and expands the plan as shown in Fig. 1f. The MODEL nodes are copied into level four in the same manner as PHANTOM nodes are copied. P1's Resolve Conflict critic notices a (CLEARTOP C) conflict between (PLAN:2a) and (PUT C ON OBJECT:1a). Resolve Conflicts inserts (SIGNAL:1a) into the plan and sends (WAIT:1a(DENY(CLEARTOP C))) to P2 and places (CLEARTOP C) on the delete list of (PLAN:2c) (Fig. 1g).

From P2's viewpoint, its planning has been completed at level three. However, when it receives the message (WAIT:1a(DENY(CLEARTOP C))) from P1, P2 must insert (WAIT:1a) before the violating action (PUT B ON C). This is shown in Fig. 2d.

At this point, both planners have completed all expansions and criticism. The plan has been completely "linearized" (synchronized) as the block movement actions:
(PUT C ON OBJECT:1a)(PUT B ON C)(PUT A ON B).

A look at the completed plans (Figs. 1g and 2d) shows that (PLAN:2a), (PLAN:2b), and (PLAN:2c) model P2's action (PUT B ON C). (PLAN:1c) models P1's action (PUT C ON OBJECT:1a), and (PLAN:1b) models (PUT A ON B). (PLAN:1a) models both (PUT C ON OBJECT:1a) and (PUT A ON B).

2.3 Distribution of NOAH's Global World Model

In the above problem, a complete and correct initial world model was assumed at both

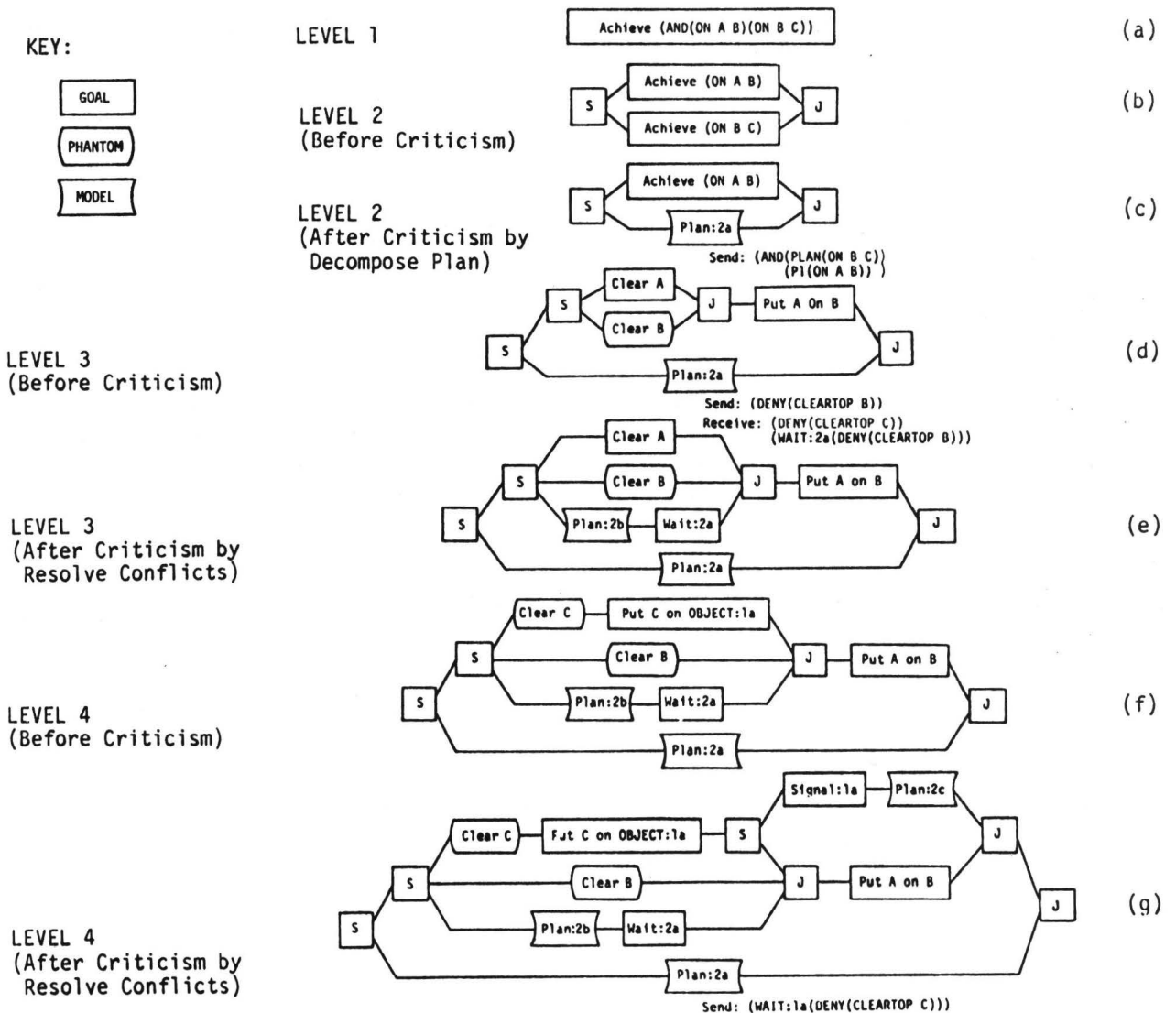


Figure 1. Three Blocks Problem: Processor P1

processors P1 and P2. These initial world models correspond to full copies of the global world model used in the centralized NOAH system. In general, only partial initial world model information may actually be needed. For example, although P1 does require the complete initial world model in the problem, P2 could successfully plan without the expression (ON C A). Use of partial initial world models opens the possibility that most of the sensory data needed to develop a local world model for each planner will be available from local sensors. Additional world model expressions, if required, would be acquired from other processors.

One approach to obtaining additional world model expressions is for each processor to announce (broadcast) relevant portions of the partial world model generated from its local sensory data to processors which are known to require this information during planning. Each processor then integrates the information it receives from other processors into its own partial world model. This integration may be as simple as taking the union of the world model expressions or as involved as a complete problem in distributed interpretation (such as described in [3]). This approach is appropriate when there is a good chance the announced expressions will be used by the receiving processor(s).

A second approach to obtaining additional initial world model expressions is to transmit requests for them. If a planner is unable to determine the value of an expression locally, it broadcasts to other processors a request for

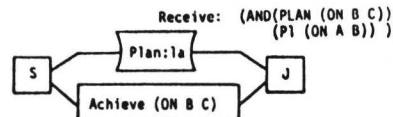
the value. A processor receiving such a request attempts to determine the expression's value, and, if successful, transmits the value to the requesting processor. This approach is appropriate when the expression is considered by the sensing processor(s) as unlikely to be used by the requesting processor.

Spatial "awareness" (knowledge) of the sensory and planning areas of the various processors is important to both approaches. In the expression announcement approach, this knowledge is used to direct expressions to processors likely to need their values and to filter those expressions unlikely to be needed by anyone. In the expression request approach, this knowledge is used to direct requests to processors likely to have the expression's value. These two uses of spatial knowledge during initial world model acquisition are analogous to the concepts of task-centered and knowledge-source-centered knowledge described by Smith [4] for indexing processing nodes and tasks in distributed systems.

2.4 Distribution of NOAH's "Distributed" World Model

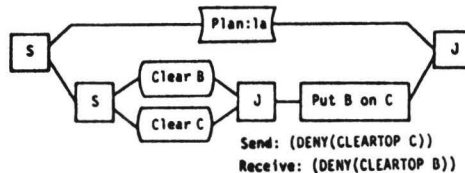
Expressions whose values are changed during local planning are removed from the local initial world model, with the add or delete list of the action indicating the new values. Denied expressions, received from other processors, are handled in a similar fashion: the expression is deleted from the local initial world model and the new value indicated on the delete list of the appropriate MODEL node.

LEVEL 2
(After Criticism by
Decompose Plan)



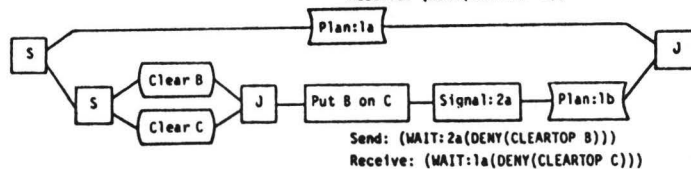
(a)

LEVEL 3
(Before Criticism)



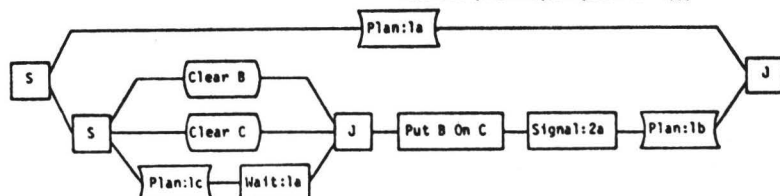
(b)

LEVEL 3
(After First Criticism
by Resolve Conflicts)



(c)

LEVEL 3
(After Second Criticism
by Resolve Conflicts)



(d)

Figure 2. Three Blocks Problem: Processor P2

Spatial awareness of the planning areas of the various processors is also important in reducing the communication of expression changes during planning. If a processor knows that the areas of direct and indirect effects of its actions and those of another processor do not interact, there is no need to communicate the effects of its actions.

Synchronization by Resolve Conflicts leads to expression ambiguity in MODEL nodes and can require additional processor interaction in its resolution. In the problem of Section 2.2, P1's Resolve Conflicts critic inserted (PLAN:2b) into the net as a result of the (WAIT:2a)/(SIGNAL:2a) synchronization. Planning then proceeded with the denied expression (CLEARTOP C) assumed to be in only (PLAN:2a). This was an incorrect assumption: the denying of (CLEARTOP C) actually precedes the (WAIT:2a)/(SIGNAL:2a) synchronization in P2's plan and therefore also belongs in (PLAN:2b). Fortunately, the incorrect assumption did not effect the need for a second synchronization because (PUT C ON OBJECT:1a) is temporally incomparable with both (PLAN:2a) and (PLAN:2b).

However, if P1 had an interacting action following the (WAIT:2a)/(SIGNAL:2a) synchronization, it would have been necessary to determine if (PLAN:2b) should, in fact, contain (CLEARTOP C) on its delete list. Without such a determination, P1 would generate an incorrect synchronization which would lead to an unnecessary double cross.

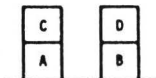
To handle these ambiguities, processors issue conditional requests for synchronization. In the problem, P1 would send the message:

```
(IF(AFTER(DENY(CLEARTOP C))
      (SIGNAL:2a))
  (WAIT:1a(DENY(CLEARTOP C)))).
```

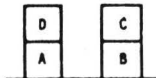
If (CLEARTOP C) is time-ordered after (SIGNAL:2a), then (WAIT:1a) is inserted by P2. Otherwise, P2 returns the message (BEFORE(CLEARTOP C)(SIGNAL:2a)). Notice that the choice between conditional and unconditional synchronization requests is made locally by P1.

2.5 Distribution of Resolve Double Crosses

A second problem illustrates implementation of a distributed Resolve Double Crosses critic. The initial state and goal state for this problem are:



Initial State:
(ON C A)
(ON D B)
(CLEARTOP C)
(CLEARTOP D)



Final State:
(AND(ABOVE D A)
(ABOVE C B))

Processor P1 is given the goal state and expands the plan to level 2 (Figs. 3a) where (ACHIEVE(ABOVE C B)) is given to P2 (Fig. 4a).

Both processors expand the plan to level 3 (Figs. 3b and 4b). P1's Resolve Conflicts

critic requests synchronization to protect the PHANTOM node (CLEARTOP D) (Fig. 3c). P2 does likewise to protect (CLEARTOP C) (Fig. 4c).

Once two temporally related (comparable) WAIT and SIGNAL actions occur in a plan, it is necessary to insure that they are not part of a double cross. Attempting to synchronize such a double cross leads to deadlock during plan execution. In this problem, P1 will not arrive at (SIGNAL:1a) because it is waiting at (WAIT:2a) (Fig. 3d). P2 will not arrive at (SIGNAL:2a) because it is waiting at (WAIT:1a) (Fig. 4d). To detect this deadlock situation during planning, P1 must determine if (WAIT:1a) is time-ordered before (SIGNAL:2a) in P2's plan. P1 already "knows" that (SIGNAL:2a) is the last action in (PLAN:2c), that it cannot execute (WAIT:1a) because of (WAIT:2a), and that no other processors can issue a (SIGNAL:1a) proceed message. If (WAIT:1a) is also in (PLAN:2c) there will be an execution deadlock.

P2 needs the analogous information about (WAIT:2a).

This ordering information is obtained by announcing the location of any wait actions that are time-ordered before a signal action. In this problem, P1 sends the message (BEFORE(WAIT:2a)(SIGNAL:1a)) and P2 sends the message (BEFORE(WAIT:1a)(SIGNAL:2a)). Once these messages are received, both processors detect the double cross (deadlock) and take steps to avoid it.

P1 can determine locally (because WAIT:2a synchronizes the use of (CLEARTOP C)) that the denying of (CLEARTOP C) formed its contribution to the double cross. As in the centralized NOAH system, an examination of variable bindings shows that (ON C A) should not be true when (PUT D ABOVE A) is executed. P1 inserts a new GOAL node, (ACHIEVE(NOT(ON C A))), into the plan in an attempt to eliminate the double cross. The (WAIT:2a) synchronization requested by P2 to protect the denying of (CLEARTOP C) is now inappropriate and is removed from the plan. The MODEL node (PLAN:2c) is merged into (PLAN:2a), and P1 sends the message (UNDENY(CLEARTOP C)) to P2. These actions are shown in Fig. 3e.

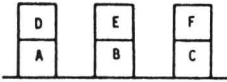
P2 proceeds similarly, as shown in Fig. 4e.

When the UNDENY messages are received, the superfluous signal actions are removed (and the appropriate MODEL nodes are merged) by both processors (Figs. 3f and 4f), and the plans are expanded to completion (Figs. 3g and 4g).

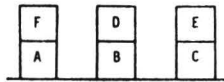
If more than two time-ordered synchronizations are required between processors, double cross (deadlock) detection may involve following chains of synchronization orderings. This may

* In the general case, deadlock detection is undecidable [5]. However, the loop- and condition-free nature of these plans allows for straightforward deadlock detection.

involve a combination of synchronization orderings and local linearization orderings. Consider the blocks world equivalent of a circular shifting of the contents of three registers:



Initial State:
(ON D A)
(ON E B)
(ON F C)
(CLEARTOP D)
(CLEARTOP E)
(CLEARTOP F)



Final State:
(AND(ABOVE F A)
(ABOVE D B)
(ABOVE E C))

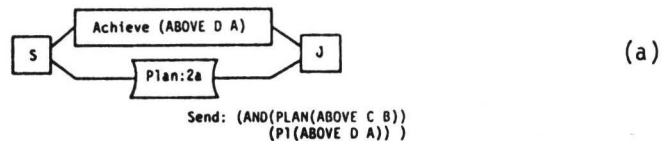
If three processors are each assigned one of the conjunctive goals, the following synchronization pattern develops:

P1: (WAIT:3a) ... (SIGNAL:1a)
P2: (WAIT:1a) ... (SIGNAL:2a)
P3: (WAIT:2a) ... (SIGNAL:3a).

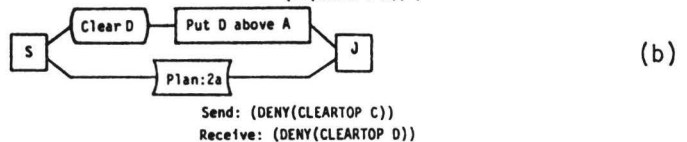
Double cross detection by P1 requires the knowledge that (WAIT:1a) is time-ordered before (SIGNAL:3a). P3 requires the knowledge that (WAIT:2a) is time-ordered before (SIGNAL:1a). This problem can be generalized to a circular shifting of the contents of N registers.

No attempt was made to modify the double cross resolution mechanisms of the centralized NOAH system. Development of more sophisticated distributed double cross resolution techniques remains an important issue.

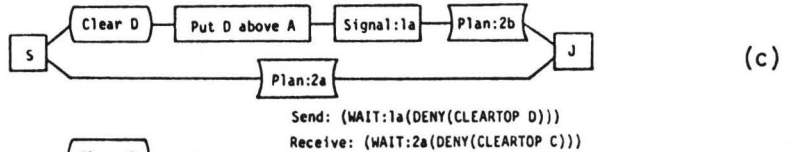
LEVEL 2
(After Criticism by
Decompose Plan)



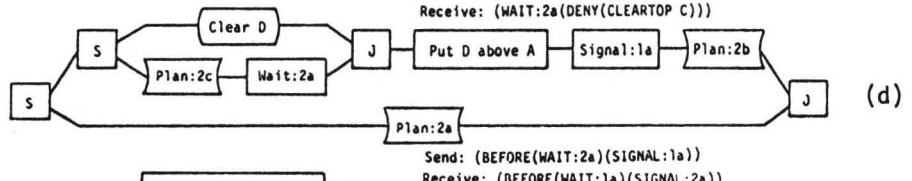
LEVEL 3
(Before Criticism)



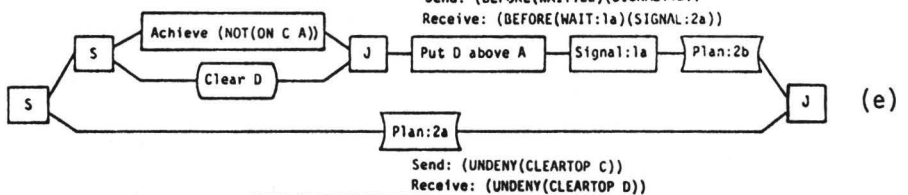
LEVEL 3
(After Criticism by
Resolve Conflicts)



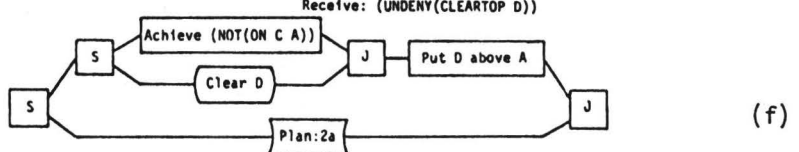
LEVEL 3
(After Second Criticism
by Resolve Conflicts)



LEVEL 3
(After First Criticism by
Resolve Double Crosses)



LEVEL 3
(After Second Criticism by
Resolve Double Crosses)



LEVEL 4

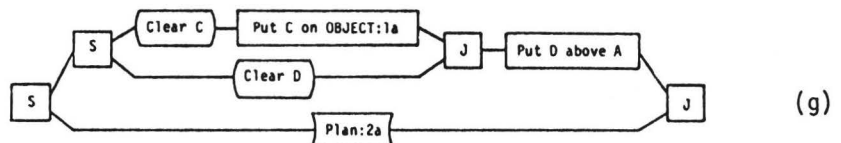


Figure 3. Swapping Blocks Problem: Processor P1

2.6 Distribution of Eliminate Redundant Preconditions

In order to detect redundant precondition achievement, it is necessary for processors to know of assertions made by other processors in a manner analogous to the use of denial information in conflict detection. Therefore, both denial and assertional messages are exchanged. PHANTOM precondition assertions, however, are not exchanged. Unless converted to GOAL nodes, interprocessor redundancies between PHANTOM nodes are not eliminated.

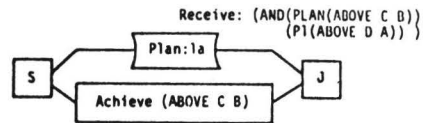
As in conflict resolution, ambiguity in the location of assertional expressions may exist and synchronization of actions may be required. A longer version of this paper [6] further describes the distributed Eliminate Redundant Preconditions critic.

2.7 Distribution of Use Existing Objects

The distributed Use Existing Objects critic analyzes all uninstantiated formal objects contained in the local portion of the plan, based on incoming assertional/denial information. If a formal object can be instantiated to achieve the same effects sent by another processor--and if the action involving the formal object can be time-ordered no later than the non-local action that achieves these effects--Use Existing Objects performs the instantiation and transmits a PHANTOMIZE request to the other processor.

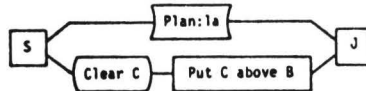
As with Eliminate Redundant Preconditions, additional synchronization may be required to insure that subsequent actions by the other processor do not proceed until the instantiated action has been executed.

LEVEL 2
(After Criticism by
Decompose Plan)



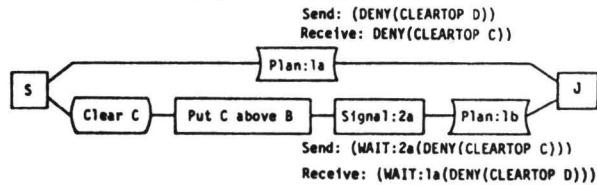
(a)

LEVEL 3
(Before Criticism)



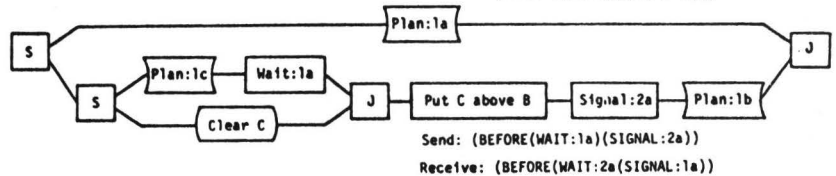
(b)

LEVEL 3
(After First Criticism
by Resolve Conflicts)



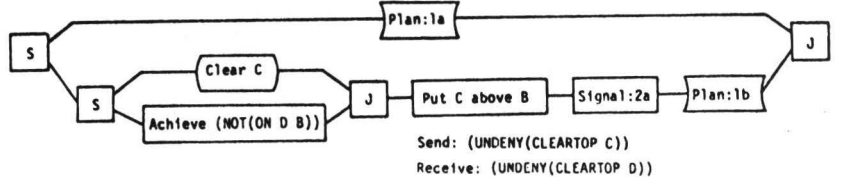
(c)

LEVEL 3
(After Second Criticism
by Resolve Conflicts)



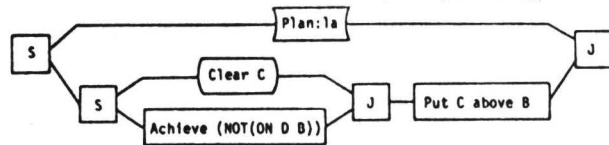
(d)

LEVEL 3
(After First Criticism by
Resolve Double Crosses)



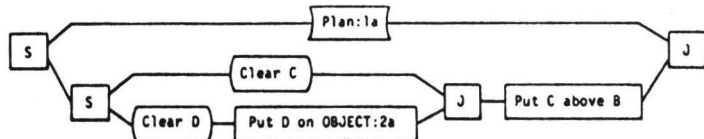
(e)

LEVEL 3
(After Second Criticism by
Resolve Double Crosses)



(f)

LEVEL 4



(g)

Figure 4. Swapping Blocks Problem: Processor P2

2.8 Additional Issues in Distributing NOAH

Integrating a distributed NOAH system into a complete distributed problem solver involves a number of additional issues: plan to processor allocation; planning versus execution-time criticism and synchronization; distributed detection of plan completion followed by plan execution versus integrated planning and plan execution; and distributed monitoring and plan modification. These issues are addressed in [2].

3. CONCLUSION

NOAH is suitable for generalization to a distributed planning system. The generalization requires introducing mechanisms for criticism and world model distribution.

ACKNOWLEDGMENTS

This paper is built on Earl Sacerdoti's milestone in planning: NOAH. The use of NOAH as a basis for distributed planning evolved during discussions with Victor Lesser and Nils Nilsson. Victor Lesser's comments on earlier drafts and during numerous discussions have been both useful and appreciated. John Lowrance, Scott Reed, and Jack Wileden also made helpful suggestions for this paper.

REFERENCES

- [1] Sacerdoti, E.D. A Structure for Plans and Behavior. New York: American Elsevier, 1977.
- [2] Corkill, D.D. "Hierarchical Distributed Planning." Technical report, COINS, Univ. of Mass., Amherst (in preparation).
- [3] Lesser, V.R., and L.D. Erman. "An Experiment in Distributed Interpretation." Technical Report CMU-CS-79-120, Computer Science Dept., Carnegie-Mellon Univ., May 1979.
- [4] Smith, R.G. "A Framework for Problem Solving in a Distributed Environment." Ph.D. thesis, technical report STAN-CS-78-700, Computer Science Dept., Stanford Univ., Dec. 1978.
- [5] Holt, R.C. "Some Deadlock Properties of Computer Systems." Computing Surveys, 4:3 (1972) 179-196.
- [6] Corkill, D.D. "Hierarchical Planning in a Distributed Environment." Technical report, COINS, Univ. of Mass., Amherst, Feb. 1979.