HIERARCHICAL PLANNING IN A DISTRIBUTED ENVIRONMENT

Daniel D. Corkill

79-14

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

February 1979

Keywords: Distributed Planning, Distributed Processing, NOAH.

ABSTRACT

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 is generally incongruous with effective distributed problem solving systems. The inappropriateness of centralized planners in most distributed applications motivates generalization of centralized planning techniques to accomodate 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. The suitability of this type of distributed planner is discussed, and a potentially more appropriate distributed planning organization is outlined.

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

1.0 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. Command and control systems, inventory control systems (car rentals, airline reservations, etc.), and tasks involving mobile robots are all examples of such applications.

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. Use of a centralized planner in these environments also invalidates such advantages of distributed systems as: increased reliability and flexibility, enhanced real-time response, lower processing cost, and the handling of complex tasks through decomposition [Lesser and Corkill, 1978]. The inappropriateness of centralized planners in most distributed applications motivates generalization of centralized planning techniques to accomodate multiple and distributed centers of planning control.

This paper describes such a generalization of Sacerdoti's NOAH (Nets of Action Hierarchies) system [Sacerdoti, 1977]. Section 2 discusses a range of possible planning decompositions for distributed environments. Section 3 discusses why NOAH is an appropriate candidate for distribution and, through blocks world problems, illustrates a technique for distributing NOAH's criticism and world model mechanisms. Section 4 highlights a major weakness of the distributed NOAH system and outlines a potentially more appropriate organization for distributed planning.

2.0 PLANNING DECOMPOSITIONS FOR DISTRIBUTED ENVIRONMENTS

As is the case with other problem solving tasks, planning can be decomposed for a distributed environment in a number of ways. One possibility is not to decompose the planning task at all. The advantages of this approach are:

1. Planning is localized to a single processor--therefore, centralized planning techniques can be used without modification.

2. Since planning is performed at a single processor, there is no interprocessor communication cost to the planning process itself.

However, use of a centralized planner has a number of disadvantages:

1. Environmental information collected by the various processors must be transmitted to the planning processor, potentially requiring a large amount of communication.

2. The completed plan must be distributed to appropriate processors, again potentially requiring a large amount of interprocessor communication.

3. There is no parallelism during the planning process.

4. Generating the entire plan at a single processor may require substantial processing capability at the planning node, forcing the need for non-uniform processing capabilities in the distributed system.

5. In order to protect the system from planning processor failure, redundant planning processors would be needed, further increasing the communication demands on the system.

At the other end of the planning task decomposition spectrum is the allocation of every subgoal to a process. These processes are then assigned to appropriate processors for execution (plan generation). This approach is in the spirit of the multiple process structures of newer AI languages (such as SAIL [Feldman, et al, 1972]) in which alternatives are explored concurrently. This approach has the following potential advantages:

1. Communication required to generate a world model is reduced because some environmental data is available locally and because planning of subgoal achievement may proceed using an appropriate partial world model.

2. The completed plan is automatically decomposed and distributed as a result of the subgoal allocation process (however, the distribution is not guaranteed to be appropriate for plan execution).

3. There is maximum parallelism during the planning process.

4. Failing processors only mean the loss of subgoals--not complete plans.

5. Local processors only plan for subgoal achievement, a less complex and processing intensive task than planning a complete plan.

The disadvantages of this approach are:

1. Techniques for coordinating this type of planning in distributed environments must be developed.

2. A distributed mechanism for intelligent subgoal allocation must be developed (the Contract Net formalism [Smith, 1978] seems an appropriate framework in which to develop a suitable subgoal allocation technique).

3. Detection and resolution of interacting or redundant actions require interprocessor communication.

These two extremes in planning task distribution have complementary advantages and disadvantages. Clearly, what is desired is an acceptable balance between world model acquisition and plan distribution costs at the one extreme, and the cost of detecting and resolving action interdependencies at the other extreme. This also involves balancing subgoal allocation, based on a priori task knowledge, with redistribution of the completed (sub)plans. These balances must be based on the spatial nature of the particular distributed application: the spatial distribution of environmental sensors, processors, and the effectors which carry out planned actions; the spatial nature of action interdependencies; and the spatial cost of communication.

A middle-ground approach is taken in distributing NOAH in which 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.

3.0 A DISTRIBUTED NOAH SYSTEM

3.1 NOAH as a Framework for Distributed Planning

NOAH is a suitable candidate for generalization to a multiple center distributed planner for several reasons:

1. Plan expansion in the centralized NOAH system 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.

2. The integration of nonlinear and hierarchical planning techniques in the centralized NOAH system reduces the combinatorial growth of planning. This reduction in the size of the planning space also potentially lowers the amount of

interprocessor communication required in the distributed system.

3. Plans generated by NOAH retain their nonlinear representation, allowing for parallel distributed execution without additional processing to detect potential parallelism.

To complete distribution of NOAH's planning phase, the world model and general criticism mechanisms must be distributed. In the following sections, we will use "classic" blocks world problems to present the distributed world model and criticism mechanisms. Although illustrative of the interactions between distributed planning elements, these problems have a high degree of interaction between actions, and the resulting plans tend to be linear in nature.

The distributed critics presented are extensions of NOAH's original Resolve Conflicts, Resolve Double Crosses, Eliminate Redundant Preconditions, and Use Existing Objects 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. In the following sections, a suitable subplan to processor allocation mechanism (a Decompose Plan critic) is assumed.

3.2 Distribution of Resolve Conflicts

To show how a distributed Resolve Conflicts critic can be implemented, we will look at a simple problem involving three blocks and two planning processors [1]. The initial state of this problem (which is assumed to be known by both processors) can be expressed with the following three assertions:

(ON C A)(CLEARTOP B)(CLEARTOP C).

The goal state of the problem is:

(AND(ON A B)(ON B C)).

These initial and goal states are illustrated in Figure 1.

A single processor, P1, is given the goal state. As in the centralized NOAH system, the top level goal action is entered on P1's local procedural net (Figure 2a) and expanded into a conjunction at the second hierarchical level (Figure 2b). Parent/child links are not included in these illustrations.

After the plan has been expanded and criticised by the other critics at the second level, P1's Decompose Plan critic allocates one of the conjuncts, (ACHIEVE(ON B C)), to a second processor, P2 (Figure 3a). The other conjunct, (ACHIEVE(ON A B)), remains P1's responsibility

----

[1] This problem, as well as others to follow, is from Sacerdoti [1977], and the blocks world SOUP procedures can be found in the appendices of that reference.

(Figure 2c). 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 [Sacerdoti, 1977], 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 [2], as shown in Figure 2d. Since (PUT A ON B) deletes the expression (CLEARTOP B), P1 must inform P2 of this change by sending the message (DENY(CLEARTOP B)).

P2 performs an analogous expansion (Figure 3b), sending the message (DENY(CLEARTOP C)) to P1 [3].

P1 receives (DENY(CLEARTOP C)) from P2 and enters it into the delete list of the MODEL node (PLAN:2a). P2 does likewise with (DENY(CLEARTOP B)), entering the expression on the delete list of (PLAN:1a).

P2's Resolve Conflicts critic notices (via its Table of Mulitple Effects mechanism [Sacerdoti, 1977]) a conflict between the denied expression (CLEARTOP B), in the MODEL node (PLAN:1a), and the PHANTOM node (CLEAR B). In effect, the expression (CLEARTOP B) is a resource that will be used by both processors during plan execution. P2 will require temporary use of (CLEARTOP B), while P1 will require permanent use (deletion) of (CLEARTOP B). Therefore, the distributed planning system must allow P2's temporary use before granting P1's deletion of (CLEARTOP B) [4]. Resolve Conflicts establishes this ordering by synchronizing the distributed conjuncts to insure that the endangered action (PUT B ON C) will be executed before the violating action (PUT A ON B) modelled in (PLAN:1a). This synchronization is performed by inserting a signalling action, (SIGNAL:2a), into the plan (Figure 3c) and sending the message (WAIT:2a(DENY(CLEARTOP B))) to P1. When P1 receives this message it inserts a wait action, (WAIT:2a), preceding

----

[2] 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. Corkill [1979a] describes planning activity without such level synchronization.

[3] Such DENY messages can be eliminated by having each processor infer (plan) 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.

[4] P1 could also "recreate" the resource (CLEARTOP B) by re-achieving the expression--in this case, a more costly approach in terms of both planning and execution.

(PUT A ON B), as shown in Figure 2e [5].

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 (Figure 3c). This node models those actions planned by P1 which will be executed following the (WAIT:2a)/(SIGNAL:2a) synchronization. The denied expression (CLEARTOP 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) (Figure 2e), 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 (CLEARTOP C) occurs in relation to the (WAIT:2a)/(SIGNAL:2a) synchronization. (PLAN:2b) represents those actions (and their associated add and delete lists) 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 (CLEARTOP 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.

P1 finds no additional criticism to perform at the third level, and expands the plan as shown in Figure 2f. 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 the endangered action (PUT C ON OBJECT:1a). Resolve Conflicts inserts (SIGNAL:1a) into the plan, sends (WAIT:1a(DENY(CLEARTOP C))) to P2, and places (CLEARTOP C) on the delete list of (PLAN:2c) (Figure 2g).

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 Figure 3d.

At this point, both planners have completed all expansions and criticism. The plan has been completely "linearized" (synchronized) as the block movement actions:

----

[5] 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.

(PUT C ON OBJECT:1a)(PUT B ON C)(PUT A ON B).
The execution sequence of the completed plan is shown in Figure 4.

A look at the completed plans (Figures 2g and 3d) 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).

3.3 Distribution of NOAH's Global World Model

In the above problem, a complete and correct inital world model was assumed at both 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 Lesser and Erman [1979]). 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 for a processor to estimate the appropriateness of initial world model expressions for the other processors. 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 [1978] for indexing processing nodes and tasks in distributed systems.

3.4 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.

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. In situations where the processors are mobile, this spatial information may be highly dynamic, changing as the various processors plan their future mobility. Determining a balance between the acquisition of processors' spatial domains versus over-estimation of the importance of world model expressions is an important design issue in these applications.

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 3.2, P1's Resolve Conflicts critic inserted an additional MODEL node (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, (WAIT:1a)/(SIGNAL:1a), in the problem because the action (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.

3.5 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 shown in Figure 5. This problem is the blocks world equivalent of exchanging the contents of two registers.

Processor P1 is given the goal state and expands the plan to level 2 (Figures 6a-c) where (ACHIEVE(ABOVE C B)) is given to P2 (Figure 7a).

Both processors expand the plan to level 3 (Figures 6d and 7b). P1's Resolve Conflicts critic requests synchronization to protect the PHANTOM node (CLEARTOP D) (Figure 6e). P2 does likewise to protect (CLEARTOP C) (Figure 7c).

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) (Figure 6f). P2 will not arrive at (SIGNAL:2a) because it is waiting at (WAIT:1a) (Figure 7d). To detect this deadlock situation during planning [6], 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

----

[6] In the general case, deadlock detection is undecidable [Holt, 1972]. However, the loop- and condition-free nature of these plans allows for straightforward deadlock detection.

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 Figure 6g.

P2 proceeds similarly, as shown in Figure 7e.

When the UNDENY messages are received, the superfluous signal actions are removed (and the appropriate MODEL nodes are merged) by both processors (Figures 6h and 7f), and the plans are expanded to completion (Figures 6i and 7g).

If more than two time-ordered synchronizations are required between processors, double cross (deadlock) detection may involve following chains of synchonization orderings. This may 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 (illustrated in Figure 8). 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.

3.6 Distribution of Eliminate Redundant Preconditions

A third problem illustrates implementation of a distributed Eliminate Redundant Preconditions critic. The initial and goal states for this problem are shown in Figure 9. Three processors are available for this problem.

Processor P1 is given the goal state and expands the plan to level 2 (Figures 10a-c) where Decompose Plan assigns (ACHIEVE(ON B C)) to P2 (Figure 11a) and (ACHIEVE(ON C D)) to P3 (Figure 12a).

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. Figures 10d, 11b, and 12b show the transmission of assertions made during level 3 expansion [7]. Notice that PHANTOM

----

[7] Additional messages, such as those needed for possible double cross detection, are omitted from these figures for clarity.

precondition assertions are not exchanged. Unless converted to GOAL nodes, interprocessor redundancies between PHANTOM nodes are not eliminated.

Planning continues at all three processors (Figures 10e, 11c-d, 12c). During this time, both P1 and P2 know from ASSERT messages that they both are planning to achieve the expression (CLEARTOP B). However, in the absence of additional information, neither can make a decision as to which processor is the globally best suited to actually achieve the expression.

After Resolve Conflicts has completed its criticism, P2 can determine that it must achieve (CLEARTOP B) before it can proceed with (PUT B ON C). No matter which MODEL node, (PLAN:1a) or (PLAN:1b), models P1's achievement of (CLEARTOP B), neither node is time-ordered prior to achievement by P2. P2 sends P1 the message (PHANTOMIZE(CLEARTOP B)) (Figure 11e), informing P1 that P2 is achieving the expression (Figure 10f).

Planning continues (Figures 10g-h, 11f-h, 12d) with a number of conflict synchronizations performed by Resolve Conflicts. It should be noted that none of the denied expression ambiguities with respect to MODEL nodes is important in determining the need for synchronization. Therefore, in this problem there is no need to issue conditional synchronization requests for conflict resolution.

There is an important ambiguity, however, in the location of the asserted expression (CLEARTOP B) in P2's MODEL nodes. In Figure 11e, if the assertion is contained in (Plan1b), the PHANTOMIZE request by P2 is correct. This is indeed the case in this problem. If, on the other hand, the assertion should only be contained in (PLAN:1a), an additional synchronization point must be established between P1 and P2 to insure P2's completion of (CLEARTOP B) achievement. Due to this ambiguity, P2 should issue the conditional request for synchronization:

```
                    (IF(BEFORE(PHANTOMIZE(CLEARTOP B))
                              (SIGNAL:2a))
                       (WAIT:2b(PHANTOMIZE(CLEARTOP B)))).
```

If (CLEARTOP B) is time-ordered before (SIGNAL:2a), then (WAIT:2b) is inserted prior to (PUT A ON B), the action requiring (CLEARTOP B) as a precondition. Otherwise, P1 returns the message:

```
                      (AFTER(PHANTOMIZE(CLEARTOP B))
                            (SIGNAL:2a)).
```

3.7 Distribution of Use Existing Objects

Implementation of a distributed Use Existing Objects critic is analogous to the implementation of the distributed Eliminate Redundant Preconditions critic. Both critics attempt to eliminate redundant GOAL nodes from distributed portions of the plan.

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.

3.8 Additional Issues in Distributing NOAH

Integrating a distributed NOAH system into a complete distributed problem solver involves a number of additional issues:

1. Plan to processor allocation.
2. Planning versus execution-time criticism and synchronization.
3. Distributed detection of plan completion followed by plan execution versus integrated planning and plan execution.
4. Distributed monitoring and plan modification.

These issues are addressed in a more detailed presentation of the Distributed NOAH system [Corkill, 1979a].

4.0 A FUNCTIONALLY ACCURATE COOPERATIVE PLANNING SYSTEM

The major weakness of the distributed NOAH system, as described in Section 3, is that planning activity is basically completely accurate, nearly-autonomous (CA/NA) in nature [Lesser and Corkill, 1979]. This type of organizational structure requires both correct and complete interactions between processors. Although the distributed NOAH system does permit some filtering of interprocessor interactions, those messages that are interchanged must be correctly received.

The Distributed NOAH system was developed by decomposing the centralized NOAH system: distributing NOAH's critics and world model. Such an approach to distributed system design often leads to the development of CA/NA distributed systems. Centralized procedures are generally designed to use complete and consistent (correct) data. In order to compensate for data incompleteness and inconsistency introduced by distribution, completeness and consistency in interprocessor interactions must be either assured or restored by additional processing.

However, by approaching the design process as one of organizing a network of individual, autonomous, processing systems into a cooperative society, it is possible to design an organization which does not require fully complete and consistent data. Such an organizational structure is called a _functionally accurate_, _cooperative_ (FA/C) distributed system [Lesser and Corkill, 1979].

Problem solving in a FA/C organizational structure takes the form of an asynchronous, incremental search process in which partial solutions are integrated to form an overall solution. Errors stemming from the use of incomplete or inconsistent information can be _implicitly_ resolved during this integration process. Such an ability to function with incomplete and inconsistent data reduces the need for synchronization and information exchange in FA/C distributed systems and potentially increases their tolerance of component failure.

Conventional planning systems require that plans be developed in a systematic fashion. These systems are basically CA/NA: the planner has complete and consistent information about the current state of the planning process (although not necessarily about the state of the world). To develop a FA/C planner, planning techniques which can generate plans through the asynchronous, incremental aggregation of incomplete and possibly inconsistent partial plans are needed.

The Hayes-Roths' cognitive model for planning [Hayes-Roth and Hayes-Roth, 1978] takes such an approach to planning. Based on a Hearsay-II architecture, their model is hierarchical, but not strictly limited to a top-down planning sequence. Instead, planning proceeds "opportunistically", with each new decision integrated into a subset of previously made decisions. New decisions also may produce independent, competing subplans at varying abstraction levels. As planning continues, some of these subplans die-out and others are merged together into larger plans. Planning terminates when an acceptable overall plan is developed.

The success of distributing the Hearsay-II speech understanding system [Lesser and Erman, 1979] suggests that the Hayes-Roth planning model can serve as a basis for the development of a FA/C distributed planning system. However, a number of issues relating to the use of partial, inconsistent plans in such a framework need to be resolved. These issues are detailed in Corkill [1979b].

5.0 CONCLUSION

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

The major weakness of the distributed NOAH system is its lack of robustness to communication loss or error. The ability to plan without complete and consistent planning data is the motivation for current work on a functionally-accurate, cooperative distributed planner.

ACKNOWLEDGMENTS

This paper is built on Earl Sacerdoti's milestone in planning: NOAH. The structure of NOAH greatly simplified the development of a distributed planning system. The use of NOAH as a basis for distributed planning evolved during discussions with Victor Lesser and Nils Nilsson. Victor Lesser has also provided a stimulating environment for research in functionally-accurate, cooperative distributed processing. His 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

Corkill, D.D. (1979a). "Hierarchical Distributed Planning," technical report, COINS, University of Massachusetts, Amherst, Massachusetts (in preparation).

Corkill, D.D. (1979b). "Cooperative Planning," Ph.D. thesis proposal, COINS, University of Massachusetts, Amherst, Massachusetts (in preparation).

Feldman, J.A., J. Low, R. Taylor, and D. Swinehart (1972). "Recent Developments in SAIL", Proceedings of the Fall Joint Computer Conference, p. 1193-1202.

Hayes-Roth, B., and F. Hayes-Roth (1978). "Cognitive Processes in Planning," technical report RAND/WN-10268-ONR, Rand Corporation, Santa Monica, California.

Holt, R.C. (1972). "Some Deadlock Properties of Computer Systems," Computing Surveys, Vol. 4, p. 179-196.

Lesser, V.R., and D.D. Corkill (1978). "Cooperative Distributed Problem Solving: a new approach for structuring distributed systems," technical report 78-7, COINS, University of Massachusetts, Amherst.

Lesser, V.R., and D.D. Corkill (1979). "The Application of Artificial Intelligence Techniques to Cooperative Distributed Processing," technical report, COINS, University of Massachusetts, Amherst, Massachusetts (a shortened version to appear in Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo).

Lesser, V.R., and L.D. Erman (1979). "An Experiment in Distributed Interpretation," Technical Report CMU-CS-79-120, Computer Science Departnemt, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Sacerdoti, E.D. (1977). A Structure for Plans and Behavior, American Elsevier, New York.

Smith, R.G. (1978). "A Framework for Problem Solving in a Distributed Processing Environment," Ph.D. thesis, technical report no. STAN-CS-78-700, Computer Science Department, Stanford University, Stanford, California.

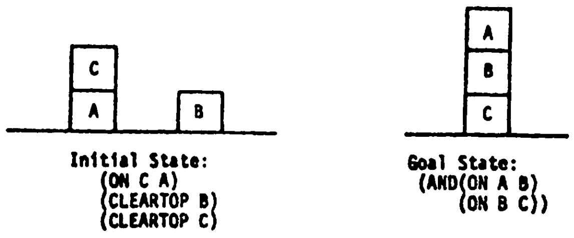


Figure 1

Three Blocks Problem

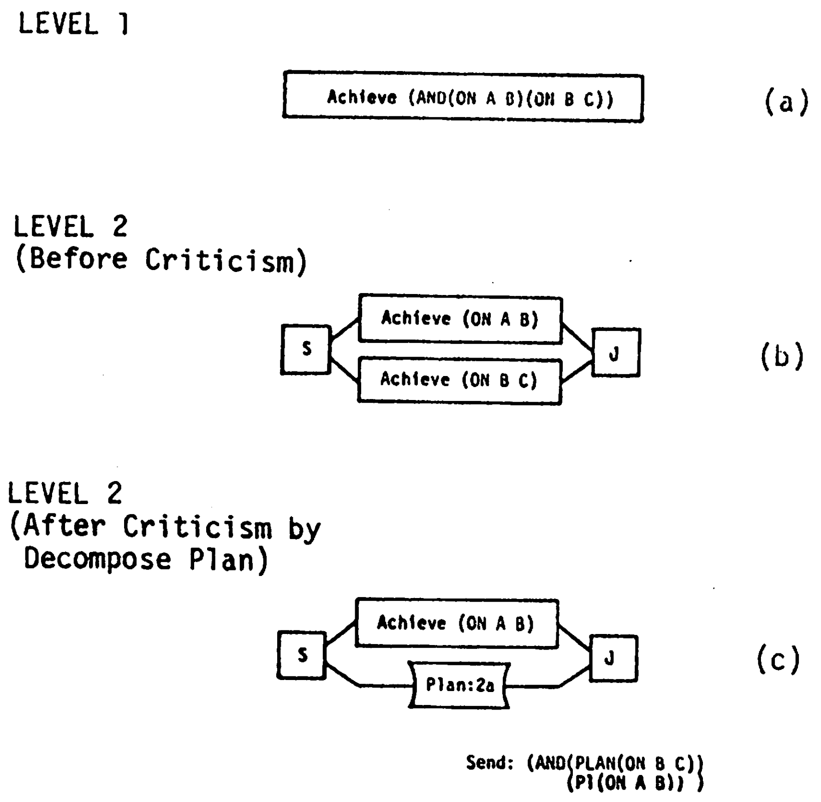
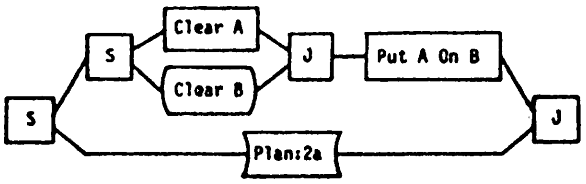


Figure 2

Three Blocks Problem: Processor P1

LEVEL 3
(Before Criticism)

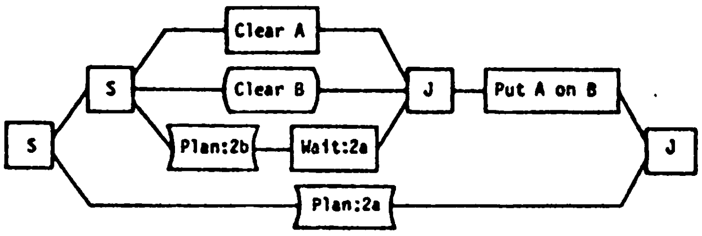


Send: (DENY(CLEARTOP B))

(d)

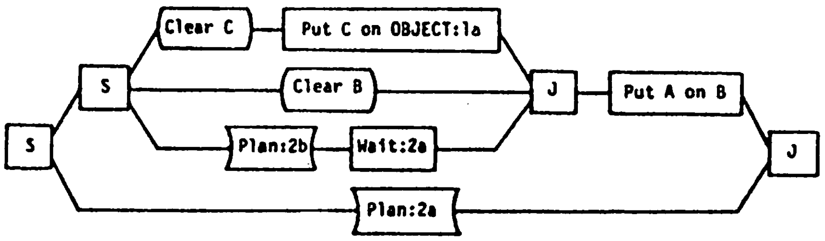
LEVEL 3
(After Criticism by
Resolve Conflicts)

Receive: (DENY(CLEARTOP C))
(WAIT:2a(DENY(CLEARTOP B)))



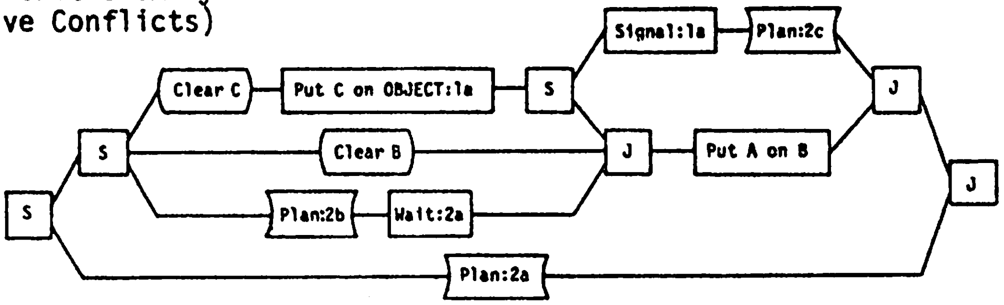
(e)

LEVEL 4
(Before Criticism)



(f)

LEVEL 4
(After Criticism by
Resolve Conflicts)



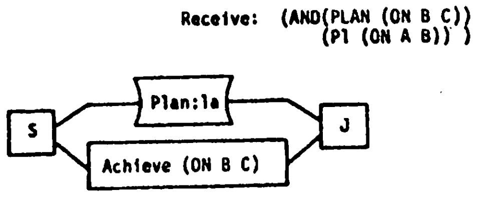
(g)

Send: (WAIT:1a(DENY(CLEARTOP C)))

Figure 2 (cont.)

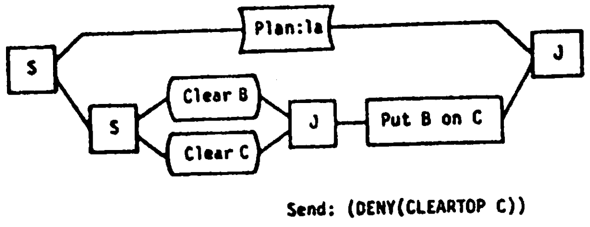
Three Blocks Problem: Processor P1

LEVEL 2
(After Criticism by
Decompose Plan)



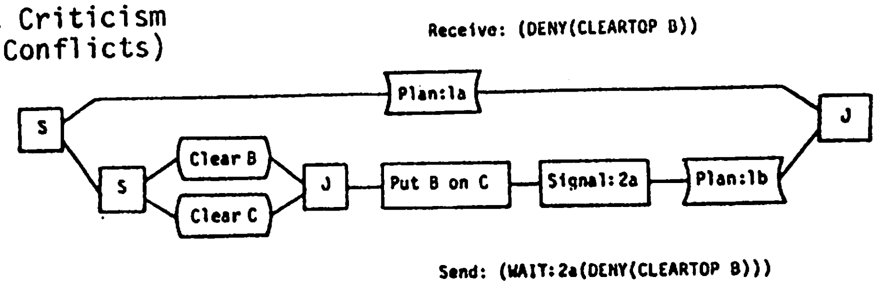
(a)

LEVEL 3
(Before Criticism)



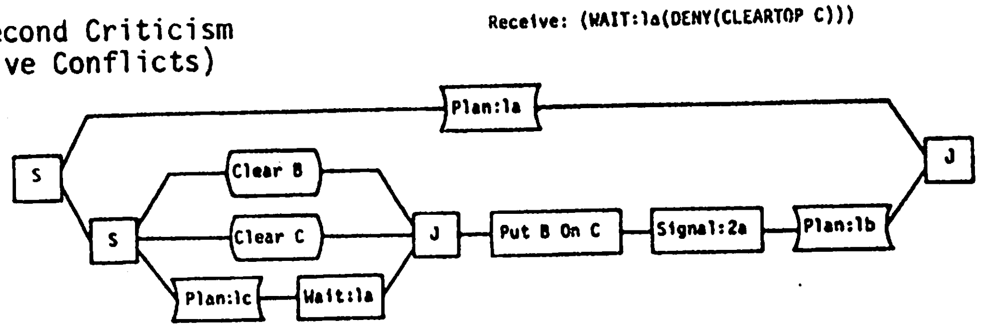
(b)

LEVEL 3
(After First Criticism
by Resolve Conflicts)


Receive: (DENY(CLEARTOP B))
Plan:1a
S
J
S
Clear B
Clear C
J
Put B on C
Signal:2a
Plan:1b
Send: (WAIT:2a(DENY(CLEARTOP B)))

(c)

LEVEL 3
(After Second Criticism
by Resolve Conflicts)



(d)

Figure 3

Three Blocks Problem: Processor P2

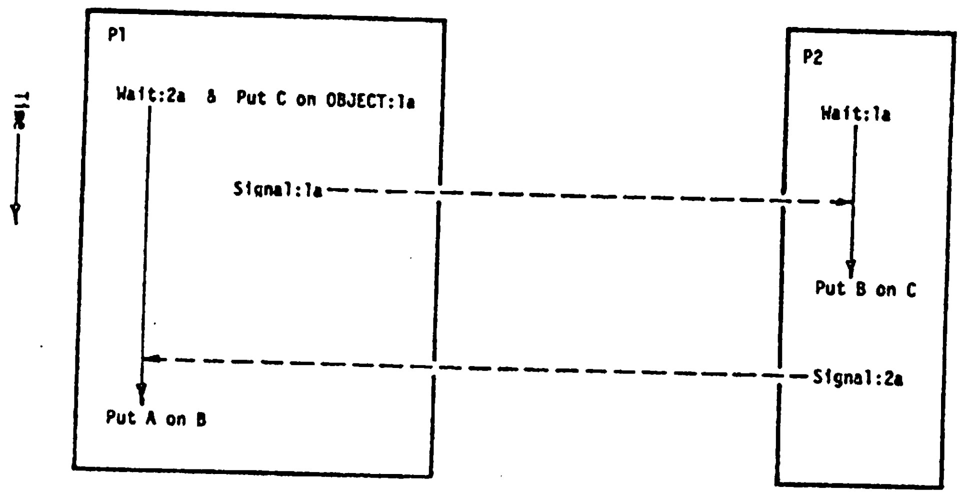


Figure 4

Three Blocks Problem: Completed Plan

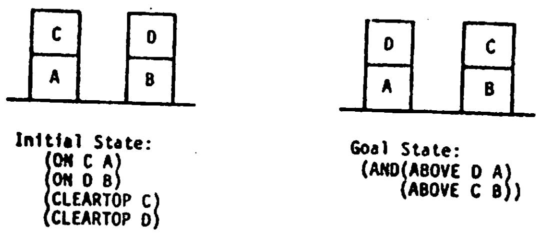
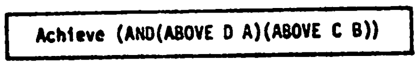


Figure 5

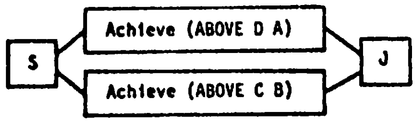
Swapping Blocks Problem

LEVEL 1



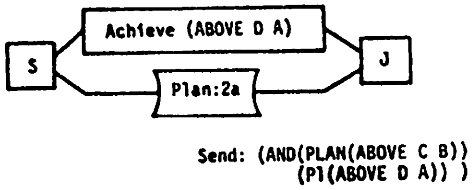
(a)

LEVEL 2
(Before Criticism)



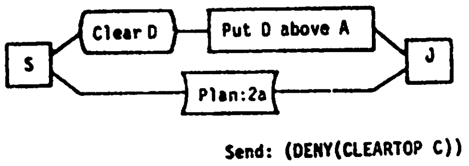
(b)

LEVEL 2
(After Criticism by
Decompose Plan)



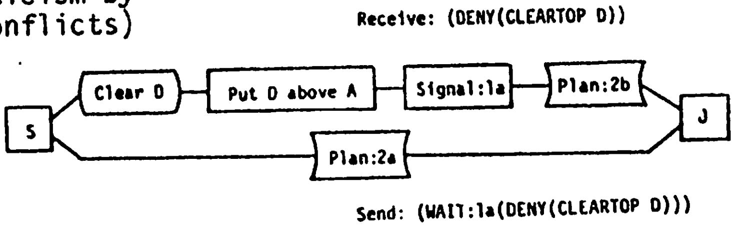
(c)

LEVEL 3
(Before Criticism)



(d)

LEVEL 3
(After Criticism by
Resolve Conflicts)

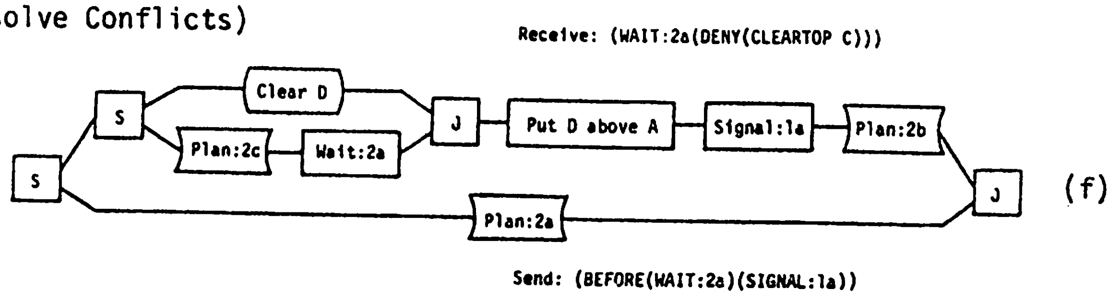

Receive: (DENY(CLEARTOP D))
S
Clear D
Put D above A
Signal:1a
Plan:2b
Plan:2a
J
Send: (WAIT:1a(DENY(CLEARTOP D)))

(e)

Figure 6

Swapping Blocks Problem: Processor P1

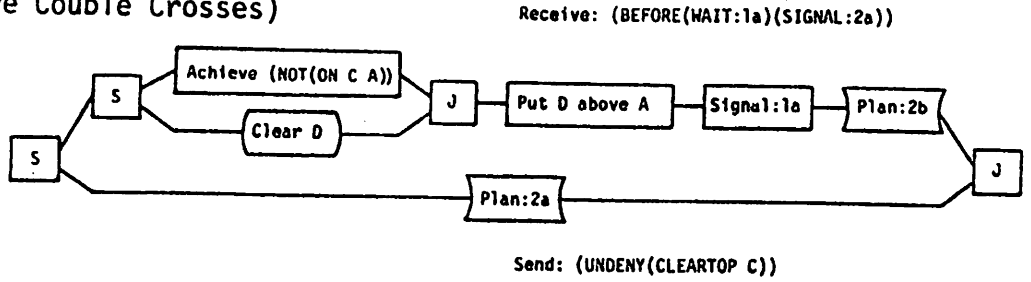
LEVEL 3
(After Second Criticism
by Resolve Conflicts)



Receive: (WAIT:2a(DENY(CLEARTOP C)))
S
S
Clear D
Plan:2c
Wait:2a
J
Put D above A
Signal:1a
Plan:2b
J
Plan:2a
Send: (BEFORE(WAIT:2a)(SIGNAL:1a))

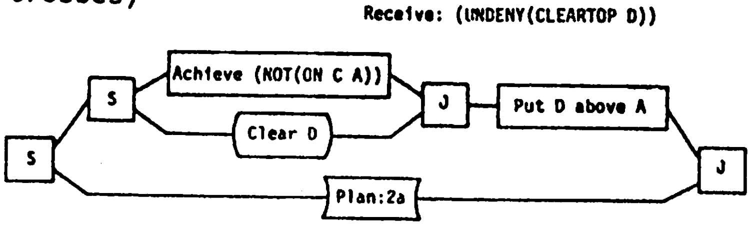
(f)

LEVEL 3
(After First Criticism by
Resolve Couble Crosses)



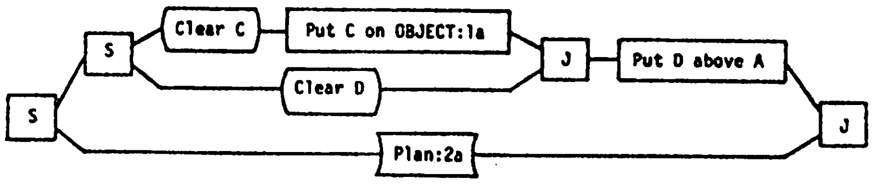
(g)

LEVEL 3
(After Second Criticism by
Resolve Double Crosses)



(h)

LEVEL 4



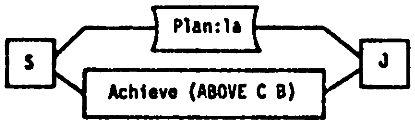
(i)

Figure 6 (cont.)

Swapping Blocks Problem: Processor P1

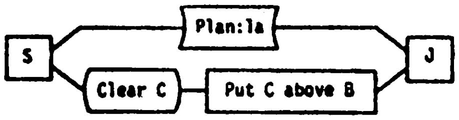
LEVEL 2
(After Criticism by
Decompose Plan)

Receive: (AND(PLAN(ABOVE C B))
(P1(ABOVE D A)))



(a)

LEVEL 3
(Before Criticism)

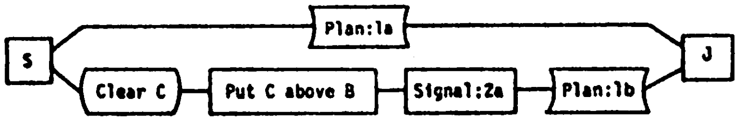


(b)

Send: (DENY(CLEARTOP D))

LEVEL 3
(After First Criticism
by Resolve Conflicts)

Receive: DENY(CLEARTOP C))

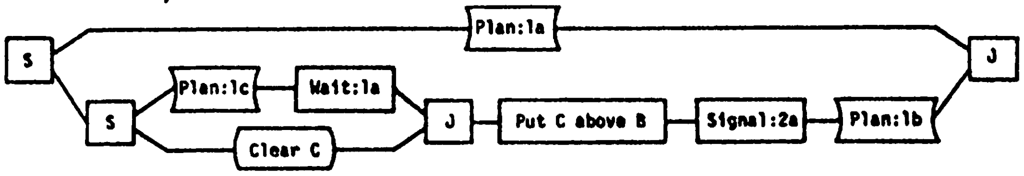


(c)

Send: (WAIT:2a(DENY(CLEARTOP C))1

LEVEL 3
(After Second Criticism
by Resolve Conflicts)

Receive: (WAIT:1a(DENY(CLEARTOP D)))

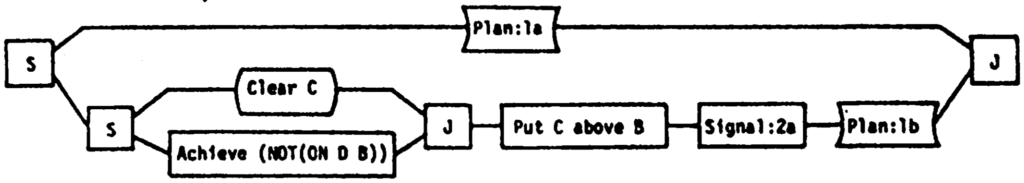


(d)

Send: (BEFORE(WAIT:1a)(SIGNAL:2a))

LEVEL 3
(After First Criticism by
Resolve Double Crosses)

Receive: (BEFORE(WAIT:2a(SIGNAL:1a))



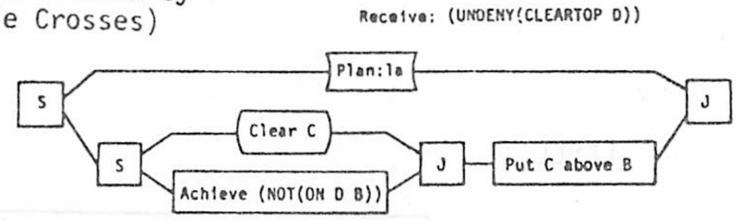
(e)

Send: (UNDENY(CLEARTOP C))

Figure 7

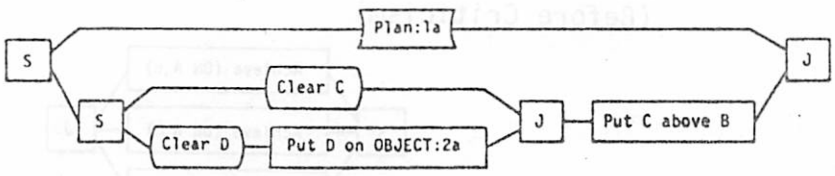
Swapping Blocks Problem: Processor P2

LEVEL 3
(After Second Criticism by
Resolve Double Crosses)



(f)

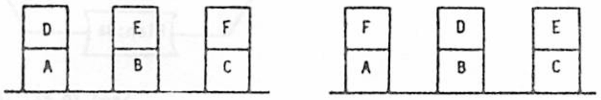
LEVEL 4



(g)

Figure 7 (cont.)

Swapping Blocks Problem: Processor P2

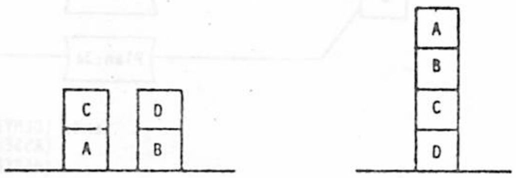


Initial State:
(ON D A)
(ON E B)
(ON F C)
(CLEARTOP D)
(CLEARTOP E)
(CLEARTOP F)

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

Figure 8

Shifting Blocks Problem



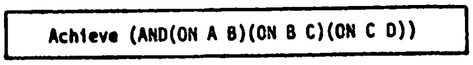
Initial State:
(ON C A)
(ON D B)
(CLEARTOP C)
(CLEARTOP D)

Goal State:
(AND(ON A B)
(ON B C)
(ON C D))

Figure 9

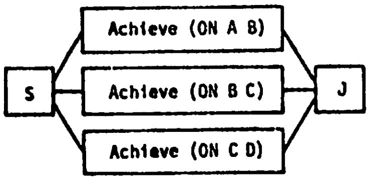
Four Blocks Problem

LEVEL 1



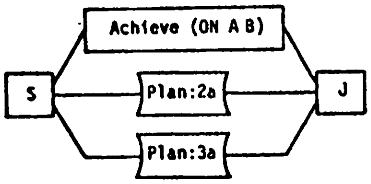
(a)

LEVEL 2
(Before Criticism)



(b)

LEVEL 2
(After Criticism by
Decompose Plan)

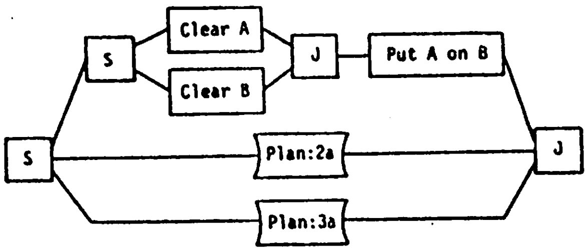


(c)

```
Send to P2: (AND(PLAN(ON B C))
                (P1(ON A B))
                (P3(ON C D)) )

Send to P3: (AND(PLAN(ON C D))
                (P1(ON A B))
                (P2(ON B C)) )
```

LEVEL 3
(Before Criticism)



(d)

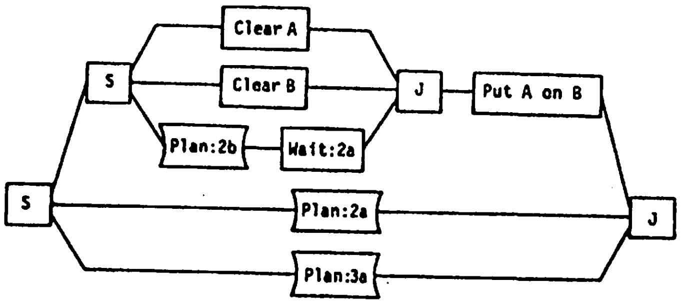
```
Send: (DENY(CLEARTOP B))
      (ASSERT(CLEARTOP A))
      (ASSERT(CLEARTOP B))
```

Figure 10

Four Blocks Problem: Processor P1

LEVEL 3
(After Criticism by
Resolve Conflicts)

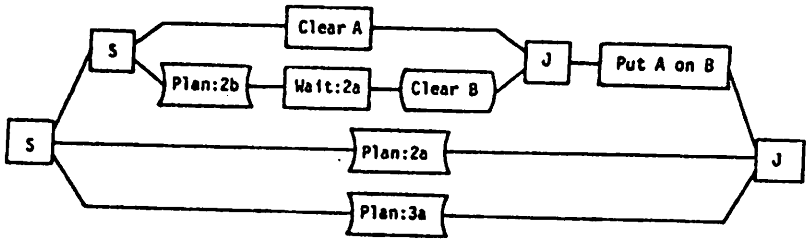
Receive from P2: (DENY(CLEARTOP C))
(ASSERT(CLEARTOP B))
(WAIT:2a(DENY(CLEARTOP B)))
Receive from P3: (DENY(CLEARTOP D))



(e)

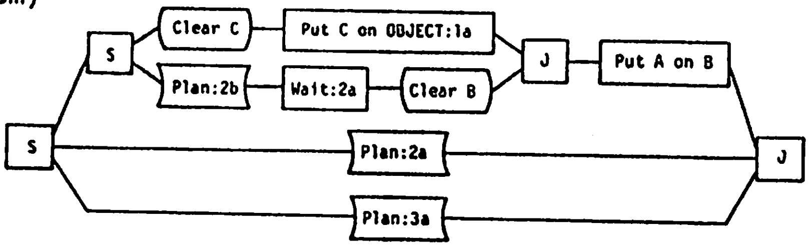
LEVEL 3
(After Criticism by
Eliminate Redundant Preconditions)

Receive from P2: (PHANTOMIZE
(CLEARTOP B))



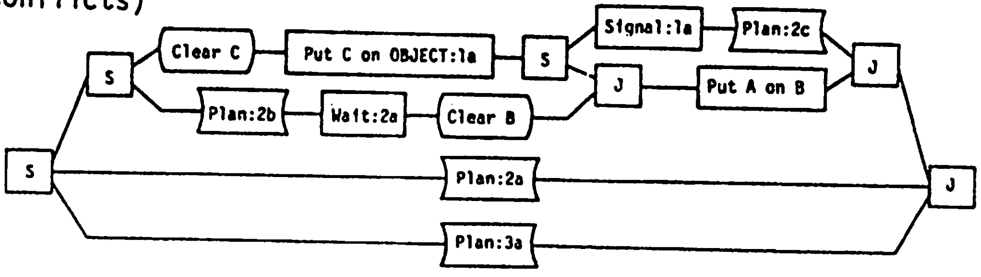
(f)

LEVEL 4
(Before Criticism)



(g)

LEVEL 4
(After Criticism by
Resolve Conflicts)



(h)

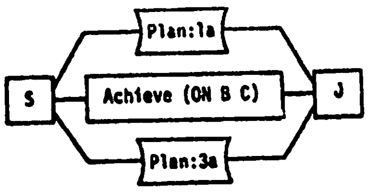
Send to P2: (WAIT:1a(DENY(CLEARTOP C)))

Figure 10 (cont.)

Four Blocks Problem: Processor P1

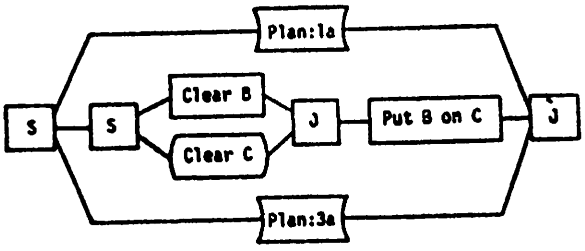
LEVEL 2
(After Criticism by
Decompose Plan)

Receive from P1: (AND(PLAN(ON B C))
(P1(ON A B))
(P3(ON C D)))



(a)

LEVEL 3
(Before Criticism)



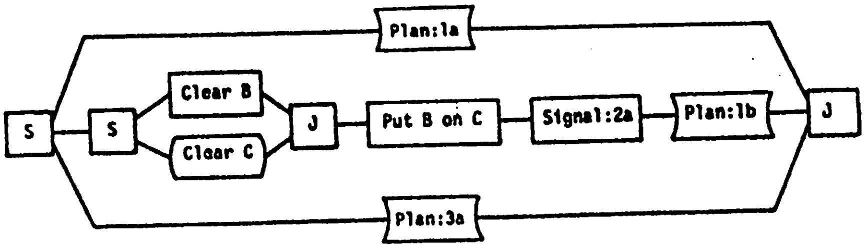
(b)

Send: (DENY(CLEARTOP C))
(ASSERT(CLEARTOP B))

Receive from P1: (DENY(CLEARTOP B))
(ASSERT(CLEARTOP A))
(ASSERT(CLEARTOP B))

Receive from P3: (DENY(CLEARTOP D))

LEVEL 3
(After First Criticism
by Resolve Conflicts)

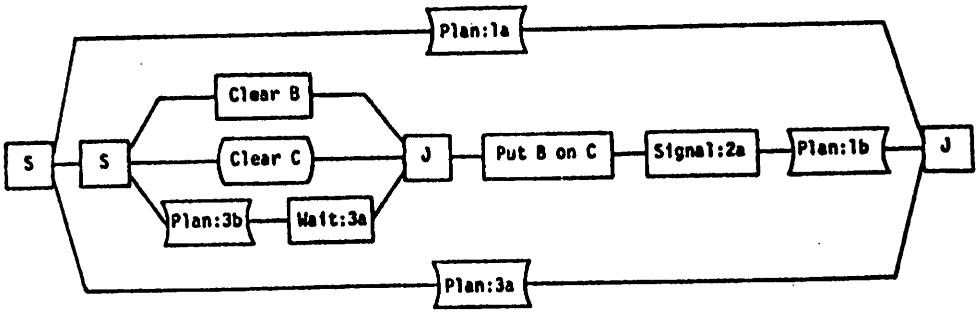


(c)

Send to P1: (WAIT:2a(DENY(CLEARTOP B)))

LEVEL 3
(After Second Criticism
by Resolve Conflicts)

Receive from P3: (WAIT:3a(DENY(CLEARTOP C)))

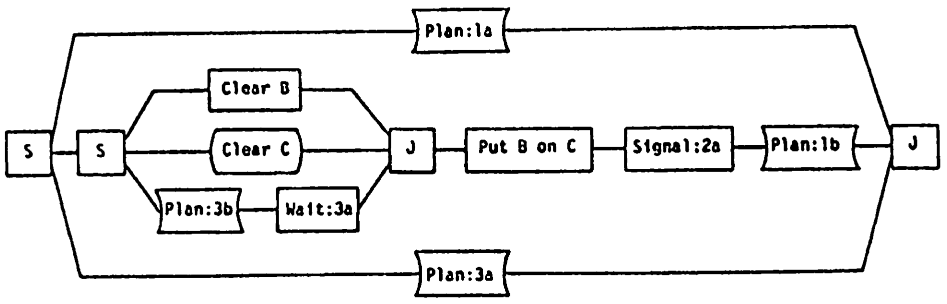


(d)

Figure 11

Four Blocks Problem: Processor P2

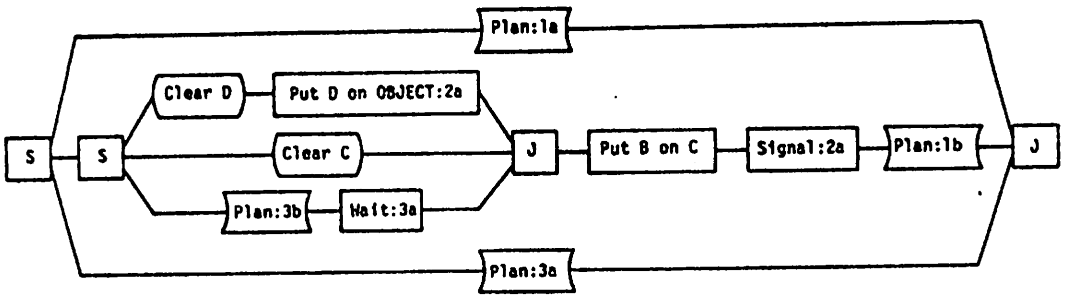
LEVEL 3
(After Criticism by
Eliminate Redundant Preconditions)



(e)

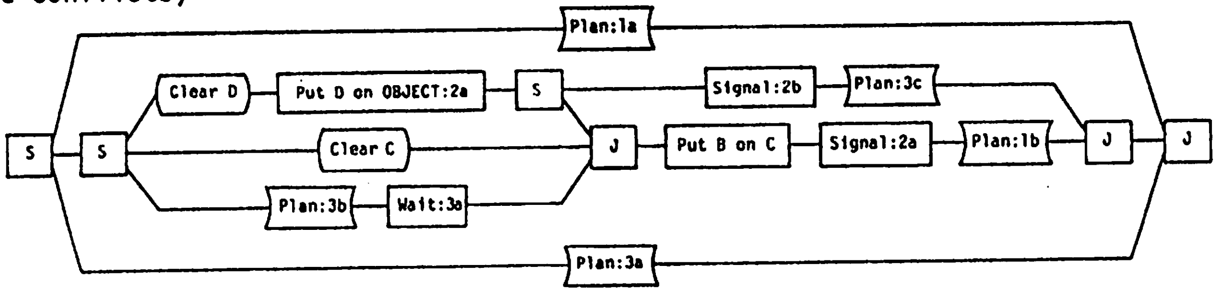
Send to P1: (PHANTOMIZE(CLEARTOP B))

LEVEL 4
(Before Criticism)



(f)

LEVEL 4
(After First Criticism by
Resolve Conflicts)

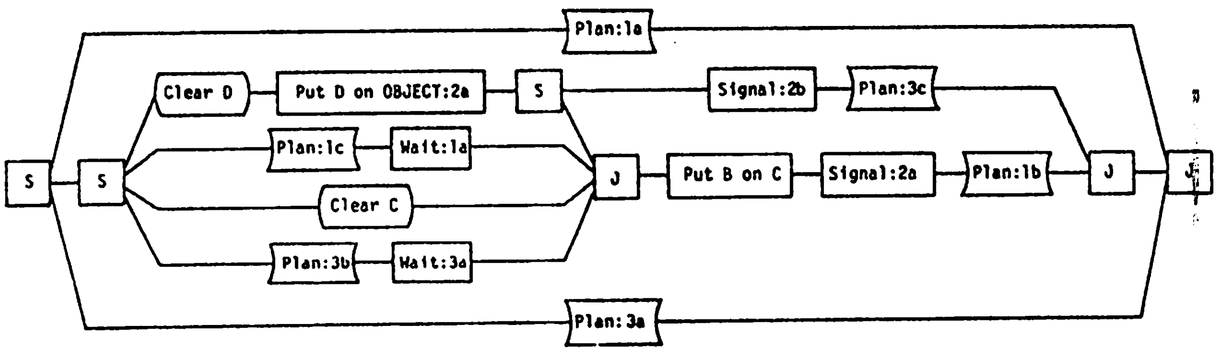


(g)

Send to P3: (WAIT:2b(DENY(CLEARTOP D)))

LEVEL 4
(After Second Criticism by
Resolve Conflicts)

Receive from P1: (WAIT:1a(DENY(CLEARTOP C)))



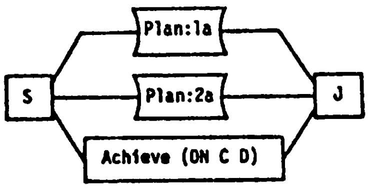
(h)

Figure 11 (cont.)

Four Blocks Problem: Processor P2

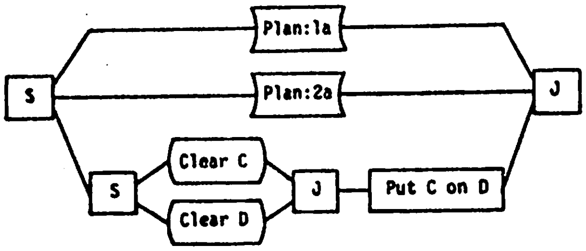
LEVEL 2
(After Criticism by
Decompose Plan)

Receive from P1: (AND(PLAN(ON C D))
(P1(ON A B))
(P2(ON B C)))



(a)

LEVEL 3
(Before Criticism)



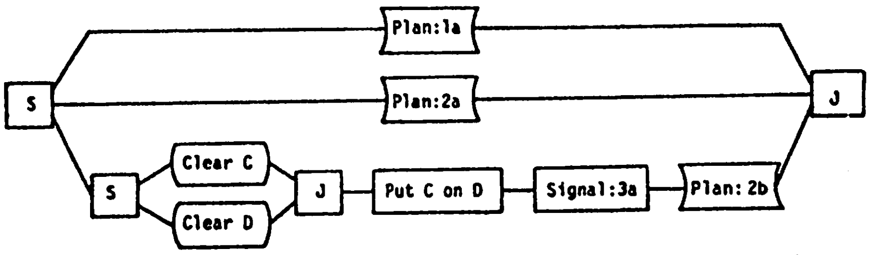
(b)

Send: (DENY(CLEARTOP D))

Receive from P1: (DENY(CLEARTOP B))
(ASSERT(CLEARTOP A))
(ASSERT(CLEARTOP B))

Receive from P2: (DENY(CLEARTOP C))
(ASSERT(CLEARTOP B))

LEVEL 3
(After First Criticism
by Resolve Conflicts)

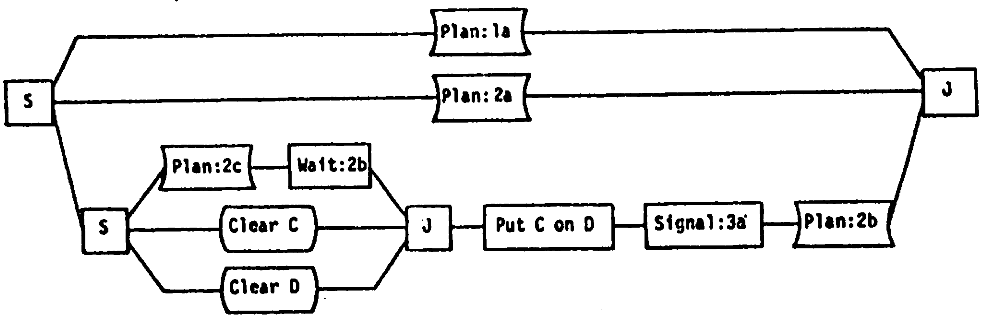


(c)

Send to P2: (WAIT:3a(DENY(CLEARTOP C)))

LEVEL 3
(After Second Criticism
by Resolve Conflicts)

Receive from P2: (WAIT:2a(DENY(CLEARTOP D)))



(d)

Figure 12

Four Blocks Problem: Processor P3

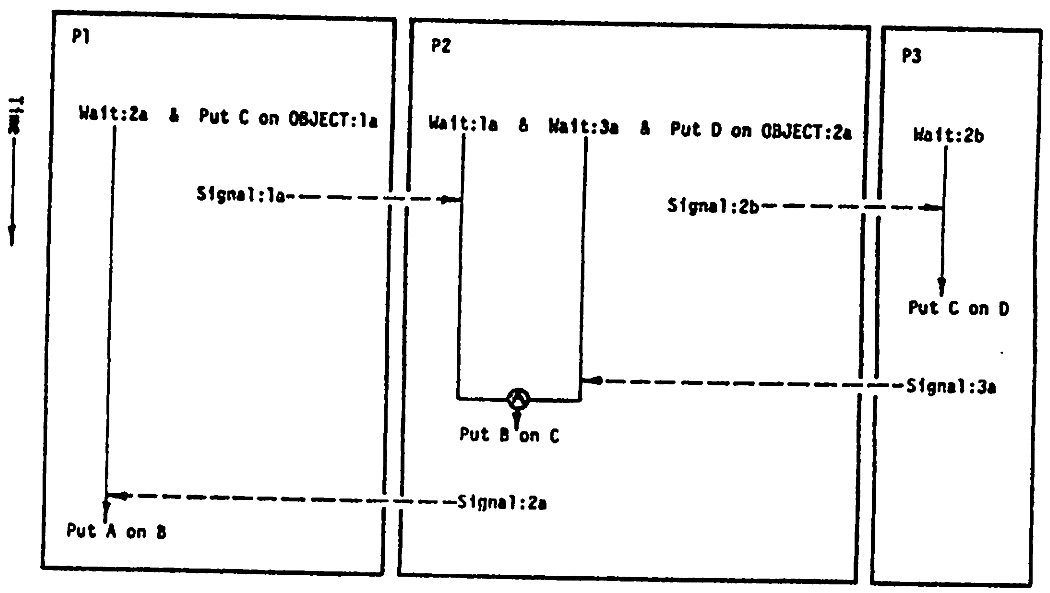


Figure 13

Four Blocks Problem: Completed Plan