A Network Representation and Algorithm for Task Planning

by

C. Zozaya-Gorostiza, C. Hendrickson

EDRC-12-08-87

December 1986

A Network Representation and Algorithm for Task Planning

by

Carlos Zozaya-Gorostiza and Chris Hendrickson

December 1986

Abstract

An activity planning algorithm based on a network representations of conditions and actions is presented. The algorithm performs backward expansions on each desired condition in order to express them as a combination of initial conditions and actions. A *global problem network* of actions and conditions is used to represent the history of successful condition expansions. This representation maintains consistency and improves the efficiency of the search procedure. Interpretation of this problem network leads to an *actions network* from which feasible plans are directly obtained. The network algorithm was implemented in LISP and is applied here to solve blockworld problems similar to those found in the artificial intellience planning literature. However, the algorithm and the network representation are general and intended to be used as an analytical tool in planning systems that formulate sequences of multiple actions.

Introduction

In a certain and closed environment, planning may be conceived in its pure form as a search problem. Given an initial state of conditions, the purpose is to find a sequence of actions that will lead to a final state of desired conditions. In a closed environment, the available operators are known in advance along with their required pre-conditions and effects. In a certain environment, the resources required by an operator and the state of the world resulting from application of an operator are predictable. Examples of such environments include blockworlds with robot manipulators [Winston 84] or control problems involving sequencing analysis tools or decision points.

In this paper, a network representation and algorithm for project planning are presented. In common with other planning models [Sacerdoti 85, Tate 77], this model involves backward expansion of desired conditions into pre-conditions through the introduction of pre-defined operators. However, in contrast to other planning models, the global problem network representation includes different nodes for conditions and operations. As a result, the *global problem network* is a bipartite graph, and efficient algorithms can be devised to obtain alternative project plans accomplishing desired goals from the global problem network of conditions and actions. These alternative plans can then be evaluated or further manipulated using standard scheduling procedures.

The network representation provides a tool for analyzing the interactions among competitive operators. It belongs to the general class of systems that formulate sequences of actions such as NOAH [Sacerdoti 85] and NONLIN [Tate 77], in contrast to those that adopt a myopic or opportunistic scheduling mechanism such as MOLGEN [Stefik 81a, Stefik 81b] and Hearsay-ll [Earman 80]. However, the network representation could also be integrated into more general planning expert systems such as OMS [Hayes 85] that repeatedly formulate useful sequences of actions within a global opportunistic heuristic planning process.

The discussion is focused on the network representation and on the plan extraction only in this paper. In the next section, some artificial intelligence (AI) planning models relevant to the planning algorithm are reviewed. The following section describes the operations used in forming the global problem network and then extracting an activity < network. The next section presents examples in a simple blockworld. A concluding section summarizes the characteristics of this planning model and contrasts it with

existing architectures.

Some AI Representation Models for Planning

NOAH

NOAH stands for *Network Of Action Hierarchies.* The system was developed after ABSTRIPS [Sacerdoti 74] and combines declarative and procedural knowledge in a framework that Sacerdoti describes as a *Procedural Net* [Sacerdoti 85, Sacerdoti 77]. Its main application has been to problems involving movements of blocks. The planning process proceeds by creating a small initial network in terms of general actions and expanding it successively until the plan has been expressed in simple operations or *primitives.* A description of the global state of the system is contained in a *world model.* The world model is modified each time the plan is detailed in order to register those changes resulting from inserting actions to the plan. Below we describe the components of the procedural net representation.

Nodes of the procedural net represent *actions.* There are five different types of nodes:

1. GOAL nodes, used to represent actions whose preconditions are false in the world model.

2. PHANTOM nodes, used to represent actions whose preconditions are already true in the world model.

3. ANDSPLIT nodes, used to indicate points of the procedural net where an action can be divided into more detailed activities that may be done in parallel.

4. ANDJOIN nodes, used to indicate points of the procedural net where an action is accomplished when all the different detailed activities have been done.

5. PLANHEAD NODES, used to indicate the beginning of a procedural net. Some changes to the world model are stored in the body of a PLANHEAD node.

A node of the procedural net may contain the following parts:

1. A verbal description of the goal to be achieved (this is only for goal nodes).

2. Preconditions that have to be true in order for the action to be performed.

3. The body of the node.

This is a piece of code that specifies a sequence of actions that when executed sequentially have the same effect as the node. This is procedural knowledge used to create a more detailed plan. When expanding a node, NOAH assumes that every action of the body contains preconditions that have to be true in order to accomplish the final action.

4. Add and Delete lists.
   Contains a list of changes to be performed to the world model. The world model is maintained using two models: a global model and a distributed model. The global model contains those expressions that have not changed since the beginning of the planning process. When a expression in the world model changes, its new value is placed in the add-delete lists of the causal node; the aggregate of such changes forms the distributed model. The original expression is stored in the PLANHEAO node.

The planning process of NOAH consists of the following three tasks:

1. Expand each node of the procedural net.
   Each node is expanded into a set of children nodes. Children actions are specified within the body code of the parent node. Those nodes whose conditions are already true in the world will be expanded as PHANTOM nodes; other children nodes are expanded by creating goal actions.

2. Apply *critics* to the new detailed procedural net.
   The main purpose of critics is to avoid conflicts and redundancies in the new procedural net. There are six types of critics in NOAH: resolve conflicts (a precondition of a node is deleted by another parallel node), use existing objects (bind formal objects only when choices are defined), eliminate redundant preconditions, resolve double-cross conflicts and optimize disjunctions.

3. Update the world model by adding and deleting objects from the world model.
   Using the add-delete lists of the new nodes in the expanded procedural net, both models of the world are maintained.

These three tasks are repeated until the plan has been decomposed into primitive actions and the plan is at the greatest detail.

NONLIN

NONLIN is a planning system developed by Tate [Tate 77] for assistance in the formulation of project networks. It is an evolution of INTERPLAN, a planning system very similar to NOAH [Tate 74, Tate 75]. Tate identified some weaknesses of NOAH and created a Task Formalism (TF) for implementing a more powerful system. TF is a purely declarative description of the actions in a hierarchical fashion and is implemented using operator schemas. Tasks are decomposed into subtasks and each subtask is assigned a *condition.* Conditions are of each of these types:

1. Unsupervised Conditions. These conditions have to exist before a task is finished, but are the responsibility of other actions.

2. Supervised Conditions. These conditions have to exist before a task is finished and are the responsibility of the action being considered.

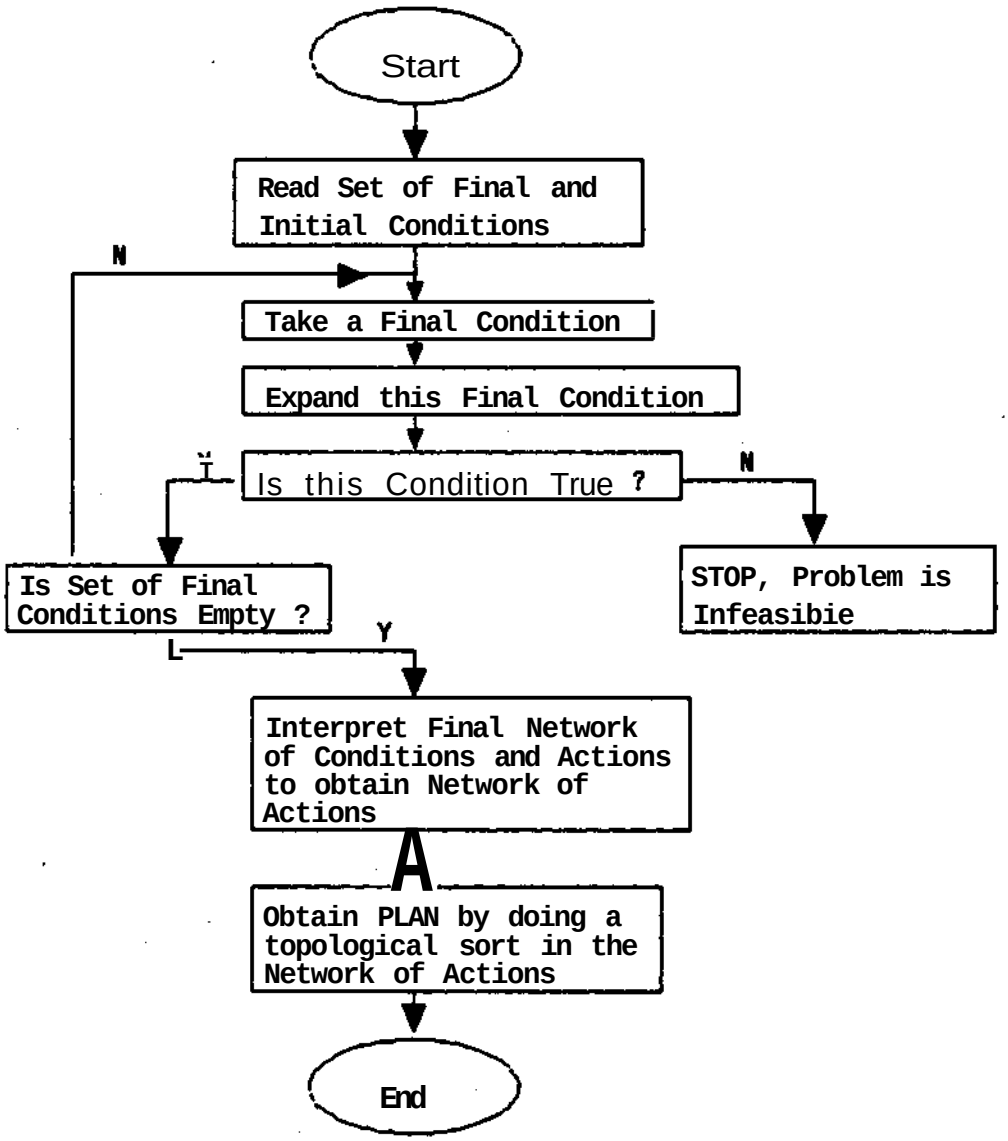
3. Use-when Conditions. These conditions are static in the sense that they do not depend on any action. However they must hold before an action is executed.

During the planning process, it is common to find that an action can be performed in two or more different ways. NOAH expanded nodes in a predetermined manner. Therefore, it required specific knowledge in the description of the nodes [Stefik 81b]. Besides, NOAH did not keep track of the choice nodes and this caused expansion decisions to be irreversible. Search trough the solution space was incomplete and some easy problems could not be solved. NONLIN solves this problem using a decision graph for backtracking when choice decisions lead to a dead end.

Nodes in Tate's network are similar to those of NOAH's Procedural Net. However, they have some other attributes in order to record interactions between each'node and its neighbors. Node components are: (1) Node number, (2) Node type (goal, action or phantom), (3) Pattern, used to look for expansion instructions, (4) Predecessor nodes, (5) Successor nodes, (6) Node context, (7) Parent node, and (8) Node mark, used to answer questions about node relations.

A Planning Model

The planning algorithm proceeds by searching *backwards* on each final condition until it is expressed in terms of actions and true conditions (figure 1). At the beginning, the user specifies the initial and final states using sets of conditions. Each final condition is expanded by adding links to a *global problem network* of conditions and actions. Once this network has been fully developed, an *activity* network is obtained. This final network represents the precedences of the required actions. One or more feasible plans may be obtained using this network. Each one is obtained doing a topological sort of the final *action network.*



Figu re 1: Overview of the Planning Algorithm

Expansion of a **condition**

The core of the planning algorithm is the expansion of false conditions into true ones. This is done by following the recursive algorithm shown in figure 2.

When a condition is expanded it is considered as an *effect* condition. The first step of the algorithm consists of identifying the set of actions that may produce the desired effect. This task is done by obtaining instances from the table of abstract *[Pre-conditions, Action, Effects]* (PAE) triplets. Each row of this table describes the preconditions and the effects of a particular action. Abstract actions are translated into concrete actions when they are bound to specific objects.

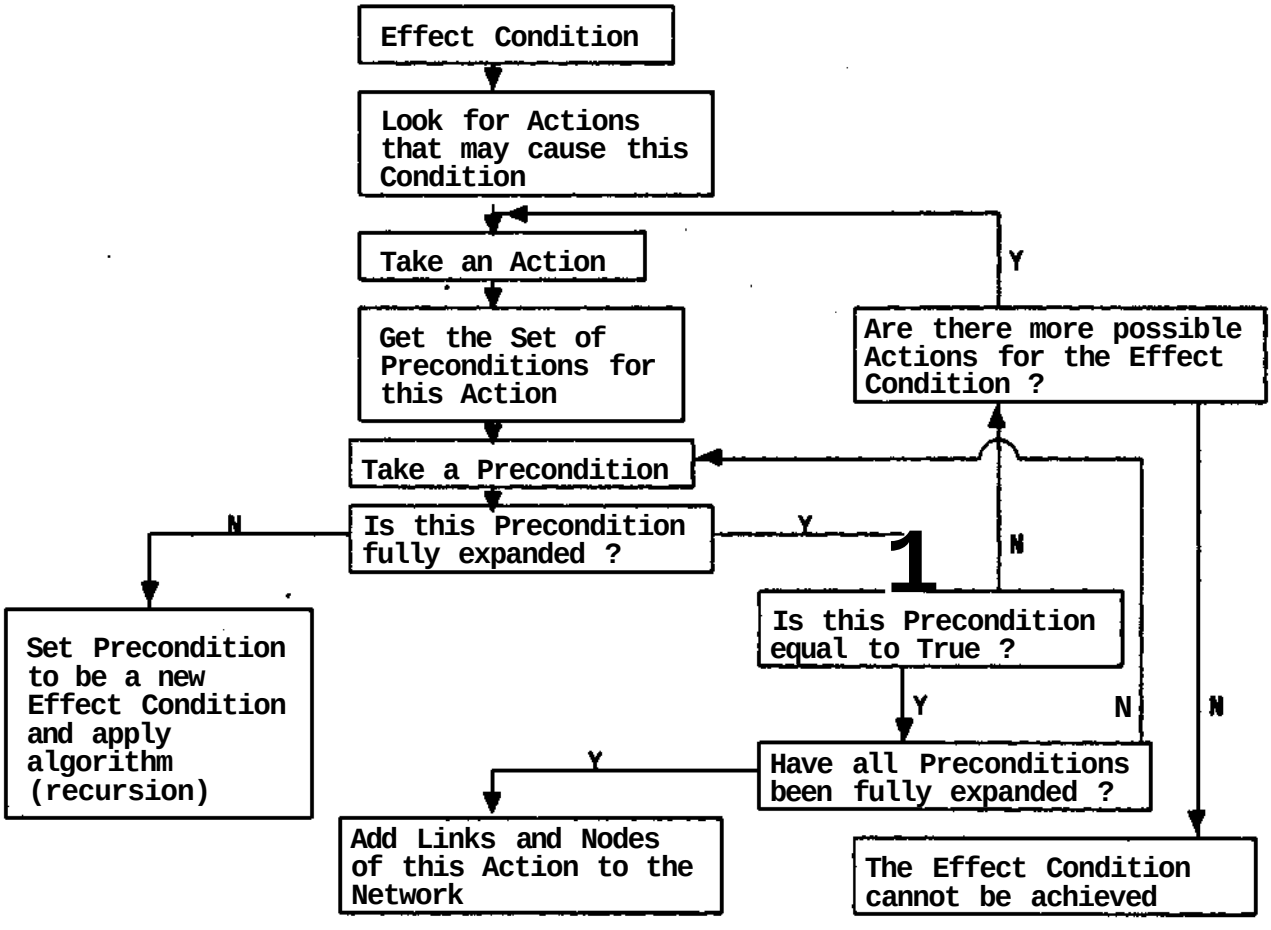


Figure 2: Algorithm for Expanding an Effect Condition

The second step of the algorithm consists of testing each one of the concrete **PAE** triplets to find a feasible action. A given action can only be accomplished if **all** its

preconditions are true. It is enough that one of them is false to discard this action. This requirement is important because it provides criteria for controlling the expansion process. When a precondition is found to be false or when the action being considered is infeasible, other actions are sought that may produce the desired effect. The algorithm is recursive because each precondition may require its own expansion before its value (True or False) is determined. Time stamps could be added to conditions in this process to represent changes in conditions over time if necessary.

Criteria for Controlling the Expansion Process

During the planning process, a condition may appear as an effect or a precondition of one or more actions. For efficiency, expansion of a condition should be done only once. This requires use of information about the status of the expansion process in order to avoid repetitions and to maintain consistency. The criteria that controls the expansion process checks first if an action is feasible or not, and then determines if there are preconditions that need to be expanded.

An action is infeasible if either:

- It negates a condition already assumed to be true in the expansion process
- If one of its preconditions is already in the expansion path

Figure 3 shows how this criteria is used. Initially, we have a set of final conditions that have to be accomplished. The set of actions that may be done for this purpose are Action 1 and Action 2. Action 1 can be done only if Conditions 1, 2 and 3 are all true. Suppose that the value of Condition 1 is not determined yet. Then we look for feasible actions to produce this effect, and we find that Actions 3 and 4 may be used. Action 3 is feasible only if neither of its preconditions (4, 5) are equal to the conditions in the path to the final condition (Final, 1). Similarly, Action 5 is feasible only if it does not negate the conditions previously defined in the tree (Final, 1, 2, 3, 4, 5) and if neither of its preconditions (8, 9) is equal to the path conditions (Final, 1, 4). Action 4 is infeasible because it negates Condition 3.

The expansion process continues until a feasible action is executable. This happens when all its preconditions are true. The value of a precondition is true if:

- It is a condition of the initial state
- It has already been expanded successfully and is part of the global network of actions and conditions

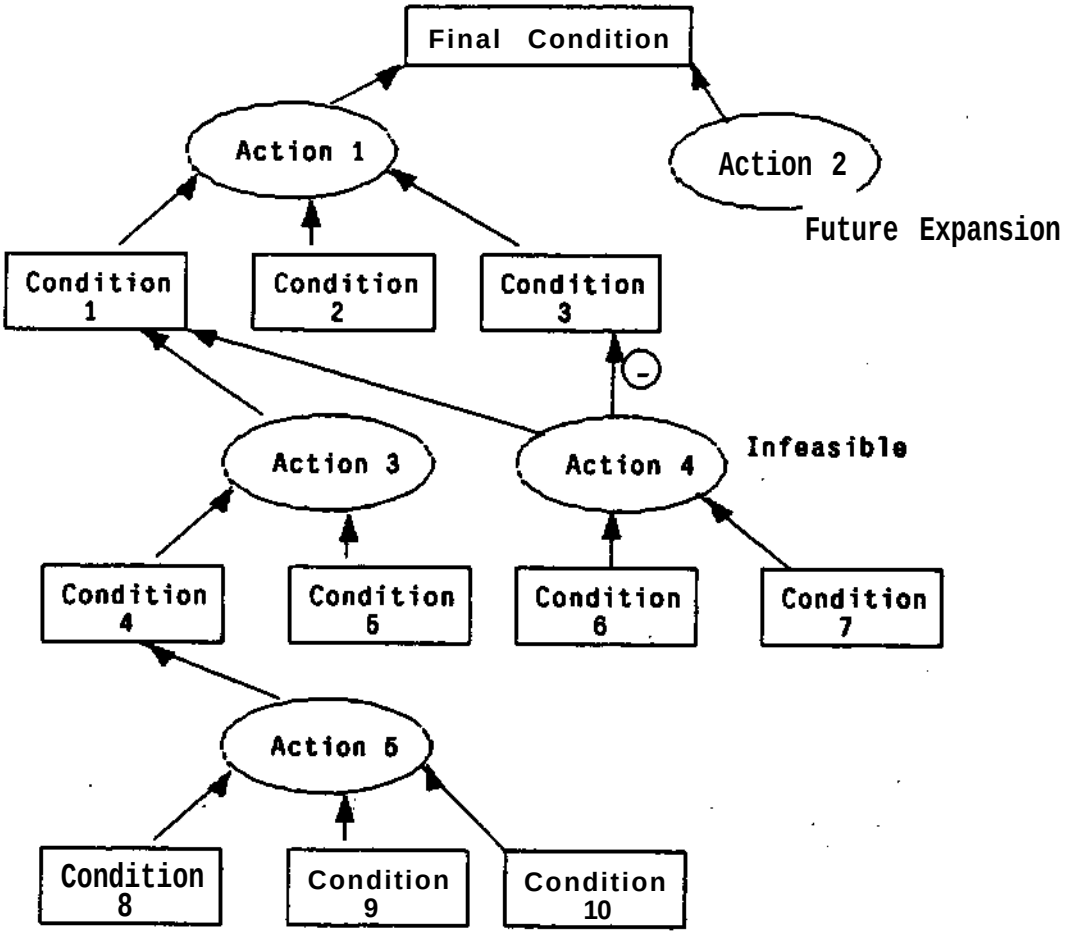


Figure 3: A Tree is Produced when Expanding a Final Condition

The value of a precondition is false if:

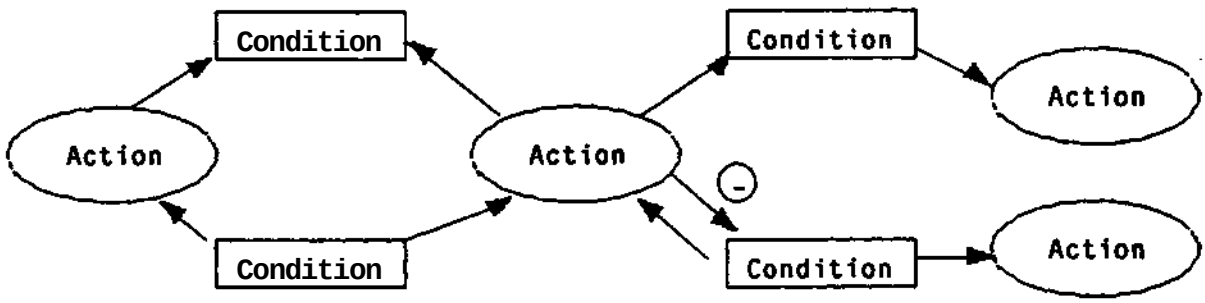
- **It has been expanded unsuccessfully. This means that there are no feasible actions that may accomplish it**

If a condition has no value, it may be expanded only if it has not already been expanded in the tree of the same final condition.

Interpretation of the Global Network of Actions and Conditions

When an action is executable, the links associated with its PAE triplet are added to the global network of the problem and the positive effects are set to true. In the global network, each action is surrounded by conditions, and each condition is surrounded by actions, as shown in Figure 4. The global network has to be translated into a new

network where the nodes are actions and the links represent precedences.



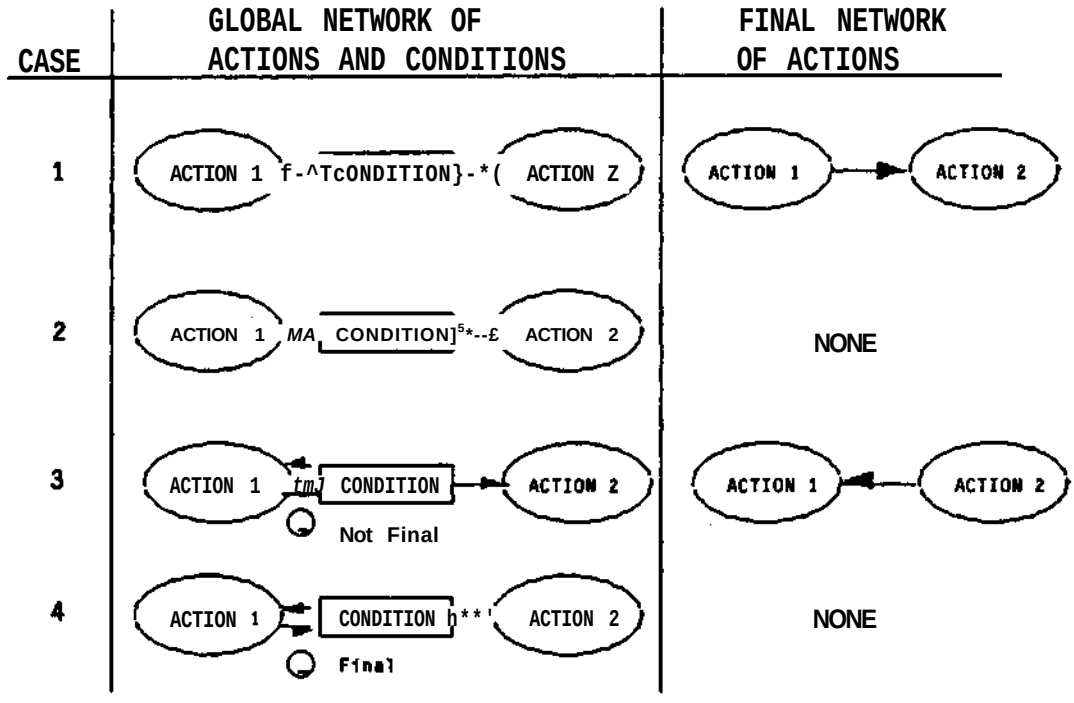
Figu re 4: Structure of the Global Network of Actions and Conditions

In order to formulate an actions-network, the relationships between neighbor actions have to be studied. The possible relations that may exist between two neighbor conditions are shown in Figure 5. Case 1 occurs when the condition between the two actions is an effect of Action 1 and a precondition of Action 2. Action 1 should be performed first in order to produce the precondition for the other action. Case 2 represents the situation where a condition is a precondition for both actions. No precedence relationship may be established in this case. Case 3 occurs when Action 1 negates a precondition of Action 2. In this case Action 2 should be done before Action 1 because if Action 1 is done first the condition would be negated and Action 2 could never be done. The exception of this situation occurs when the condition is a final condition. Given that there is at most one node in the global network for each final condition, a third action is required in order to reestablish the final condition after action 1 has negated it. Action 2 can be done then before or after action 1.

Examples of the Planning Algorithm in the Blockworld Domain

Introduction

Consider the problem of having a set of blocks in a certain initial position and wanting to move them to a different position. We want to know what sequence of actions we should perform in order to achieve the desired position. There are rules about the type and applicability of operators that constrain the set of actions that may be performed at each step of the planning process. This problem is commonly known as a *blocks world* problem. Further, suppose that we have that all the blocks are of equal shape and that



Figu re 5: Obtaining Precedence Relationships among Neighbor Actions

they have a cubic form. We describe a certain position of the blocks with only two types of *conditions:*

LCleartopX
This condition is true if there is no other block on top of block X.
2.0nXY
This condition is true if block X is on top of block Y.

For example, Rgure 6 shows drawings of several possible accommodations of the blocks A, B and C, with their respective description using the two types of conditions. In this simplified problem, no distinction is made with respect to left or right (horizontal axis). The table where blocks are located is assumed to be large enough to accommodate all blocks in a horizontal position.

Once the conditions have been specified, the set of possible actions needs to be identified. Again, we will simplify our problem to include only two types of actions:

1. Clear X
Take block on top of block X and put it on the table.
ZPutonXY
Put block X on top of block Y.

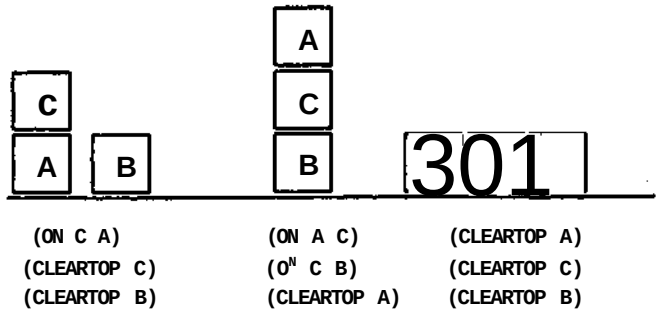


Figure 6: Two types of conditions are used to describe the positions of the blocks

The next step is to interrelate actions and conditions. We identify a set of *preconditions* and *effects* for each action. In our example, we will consider the following precondition-action-effect (PAE) triplets:

1. Action: Clear X
   Preconditions: There must be one and only one block on top of block X (say that this block is block Y).
   Effects: Reset to false the condition Y is on top of X and indicate that X is now clear on the top.

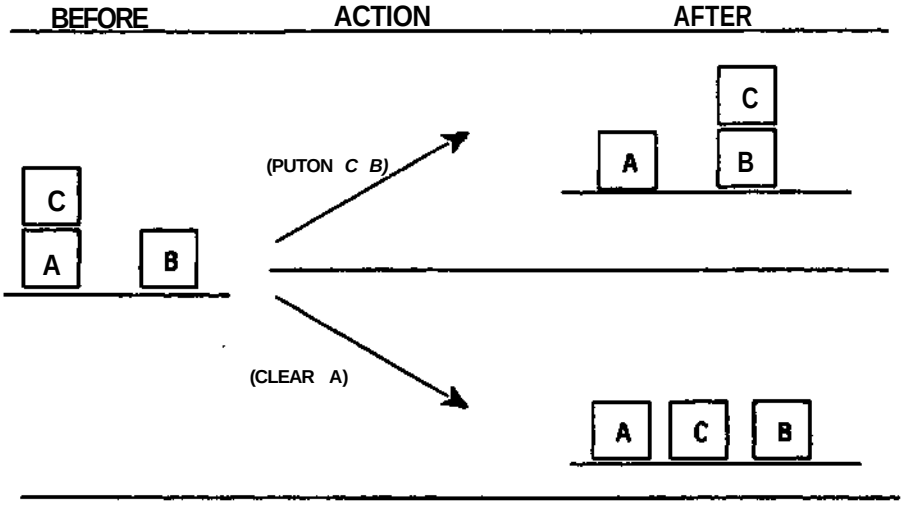
2. Action: Put on X Y
   Preconditions: Both blocks have to be clear on the top. We are not constraining X to be on the table.
   Effects: Y is no longer clear on the top, because block X is now on top of block Y.

Figure 7 summarizes these PAE triplets.

Preconditions	Action	Effects
(ON Y X) (CLEARTOP Y)	(CLEAR X)	NOT (ON Y X) (CLEARTOP X)
(CLEARTOP X) (CLEARTOP Y)	(PUTON X Y)	NOT (CLEARTOP Y) (ON Y X)

Figure 7: Table of abstract Preconditions-Action-Effects triplets for blocks world problems

Figure 8 presents the results of applying two different operators to the same *state* of the world. The set of operators that may be applied to a particular state of the world depends upon the true conditions that describe this state.



Figu re 8: Changing from one state of the world to others using actions

First Example: A Three-Blocks Problem

The problem is stated as follows:

With Initial Conditions	Achieve
(On C A)	(On A B)
(Cleartop C)	(On B C)
(Cleartop B)	(Cleartop A)

The algorithm starts by expanding the final condition (On A B), as shown in Figure 9. Given that this condition is not in the global network and that it is not an initial condition, we look into the table of abstract PAE triplets for possible actions. The only possible action is (Puton A B), with precondition (Cleartop A) and (Cleartop B). Condition (Cleartop A) is not an initial condition and is not in the network, therefore we expand it. There are two possible actions: 1- Action (Clear A), with preconditions (Cleartop B) and (On B A) of which one is already satisfied; and 2- Action (Clear A), with preconditions (Cleartop C) and (On C A), both of which are initial conditions. This second action is chosen, and the enclosed tree is joined to the global network.

Once the final condition (On A B) has been satisfied, the next final condition (On B C) is

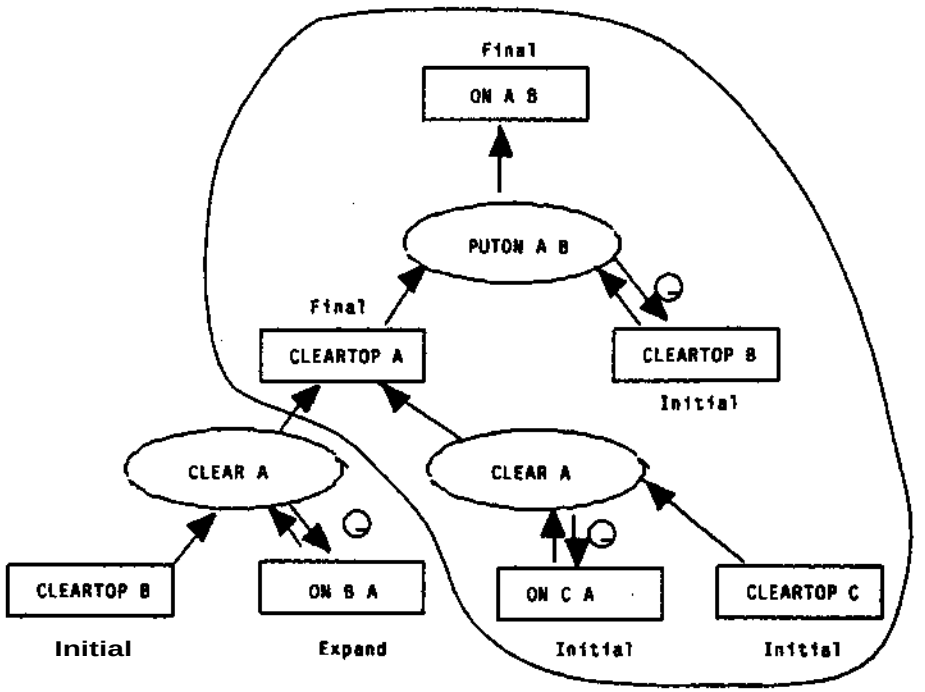


Figure 9: Expansion Tree of Final Condition (On A B)

expanded. There is only one possible action: (Puton B C), with preconditions (Cleartop B) artd (Cleartop C), both of which are true. This tree is added to the global network. The third final condition (Cleartop A) is already in the global network, therefore it is not expanded again. Figure 10 shows the global network for the problem.

The next step is to interpret this network. Starting with action (Clear A) we see that its effect (Cleartop A) forms the preconditions of action (Puton A B). According to case 1 of Figure 5, (Clear A) should be done before (Puton A B). Continuing with action (Puton B C) we see that it negates the condition (Cleartop C), a precondition of (Clear A). Using case 3 of Figure 5 (Clear A) has to be done before (Puton B C). Finally, (Puton A B) negates (Cleartop B), a precondition of (Puton B C). Using case 3, (Puton B C) has to be done before (Puton A B). Figure 11 shows the actions-network for this problem. The only feasible plan is (Clear A), (Puton B C), (Puton A B).

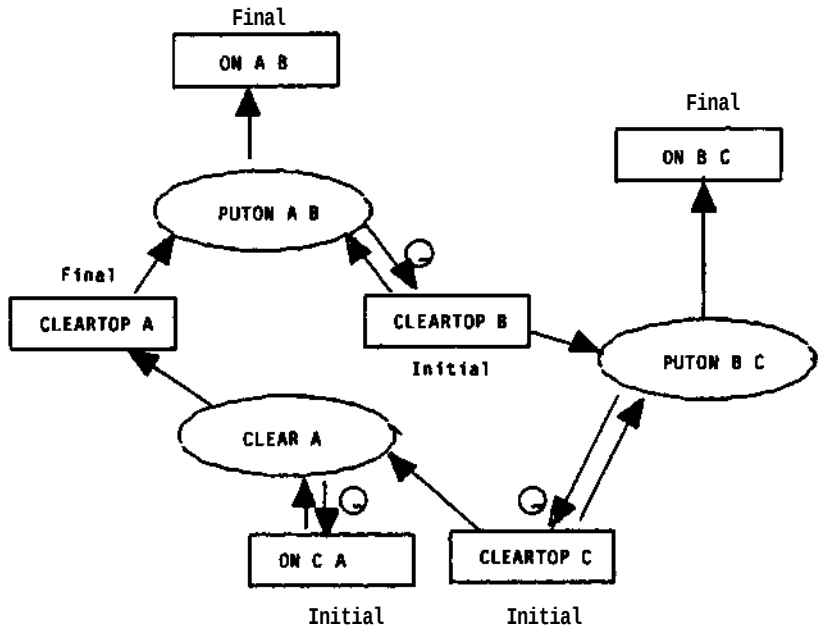


Figure 10: Global Network for the Three-Blocks Problem

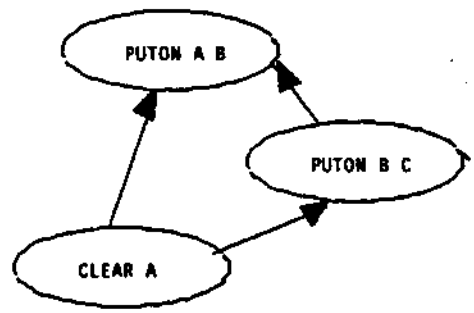


Figure 11: Actions-Network for the Three-Blocks Problem

An Example with Double-Cross Conflicts

Corkill [Corkill 79] refers to the problem shown in Figure 12 as an example with double-cross conflicts. He proposed to work with two parallel processors that would send messages to each other indicating when they may proceed and when they should stop. Corkill's work represents an extension to Sacerdoti's NOAH.

When we solve this example using the network algorithm^we find the global network of actions and conditions shown in Figure 13. The corresponding actions-network is shown in Figure 14. A topological sort of this network reveals that there are multiple possible plans as long as the precedences are not violated. For example, we could do (Clear A)

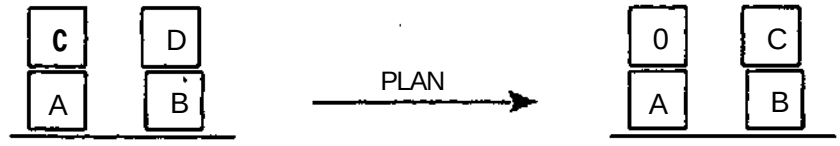


Figure 12: Example with Double-Cross Conflicts

and (Clear B) in parallel and then (Puton C B) and (Puton D A). One might argue that either (Clear A) or (Clear B) is an extra action, because the action (Puton X Y) leaves also the block below X clear on the top. However, this effect was not included in the table of abstract PAE's and this is why both Clear actions appear in the plan.

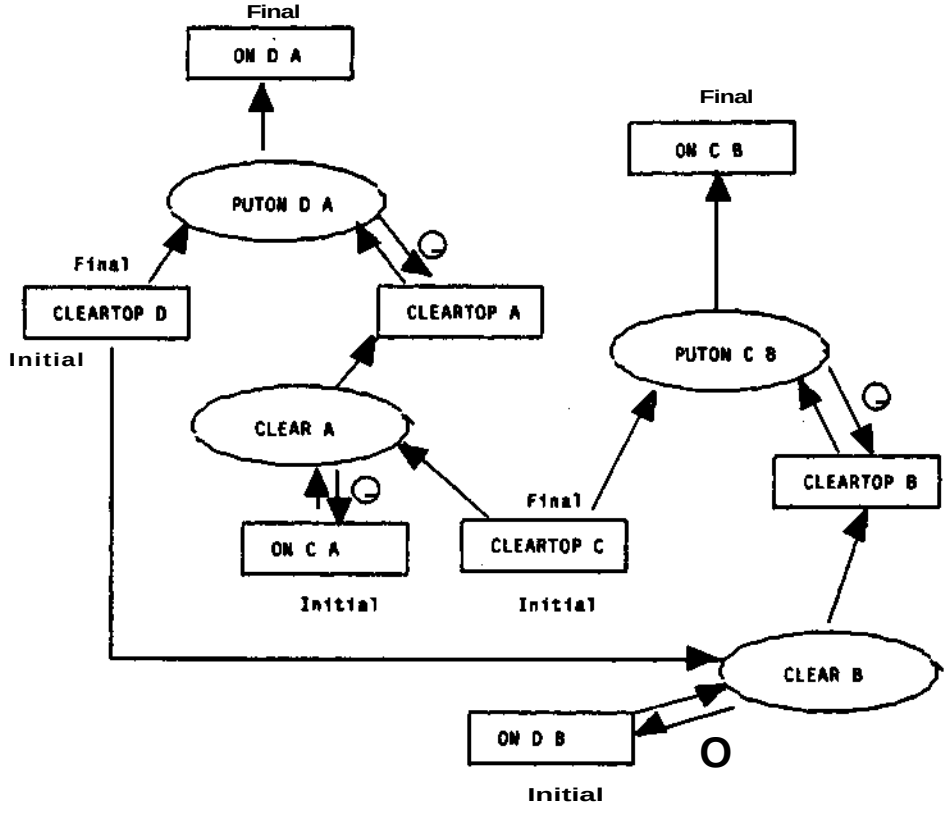


Figure 13: Global Network for the Example with Double-Cross Conflicts

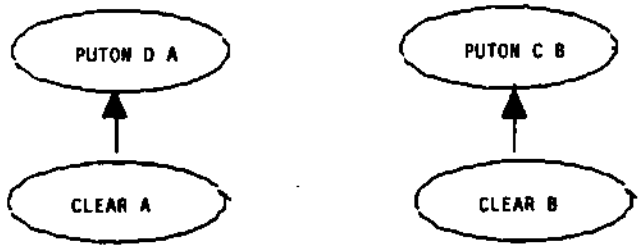


Figure 14: Actions-Network for the Example with Double-Cross Conflicts

An Infeasible Example

Suppose that we want to solve the following problem:

With Initial Conditions	Achieve
(Cleartop A)	(On A B)
(Cleartop B)	(On B A)

This problem is infeasible because either block A is on the top of block B or Block B is on top of block A. This example produces the actions-conditions network shown in Figure 15. Interpretation of this network leads to the final actions-network shown in Figure 16. Topological sort of this network is impossible, because there are cycles in the network. Therefore the problem cannot be solved.

A Six-Blocks **Problem**

In order to illustrate the application of the planning algorithm to problems with more that four blocks, the example shown in Figure 17 was solved. Initially, this problem was given to a forward-search program that was unable to find a successful plan. After 100 node expansions, early v/rong decisions had not yet been corrected.

Figure 18 shows the global network of actions and conditions for the problem. This network is a small part of the network with all possible combinations that could be obtained.

Figure 19 shows the actions-network for the problem. Again, the action (Clear F) is introduced because the side-effect (Cleartop Z) of the general action (Puton X Y) was not included in the table of abstract PAE's triplets. With this restriction, the actions-network contains the minimum number of actions that have to be **done.**

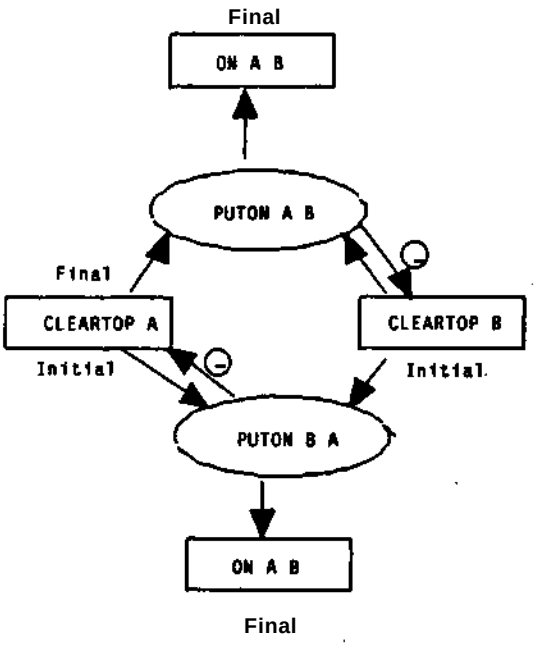


Figure 15: Global Network with Actions and Conditions for the Infeasible Example

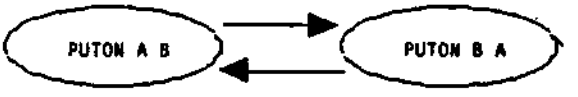


Figure 16: Actions Network for the Infeasible Example

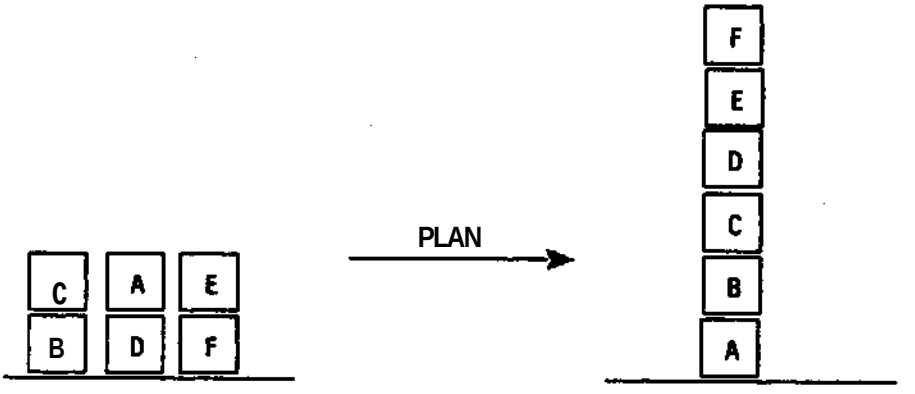


Figure 17: Example with Six Blocks

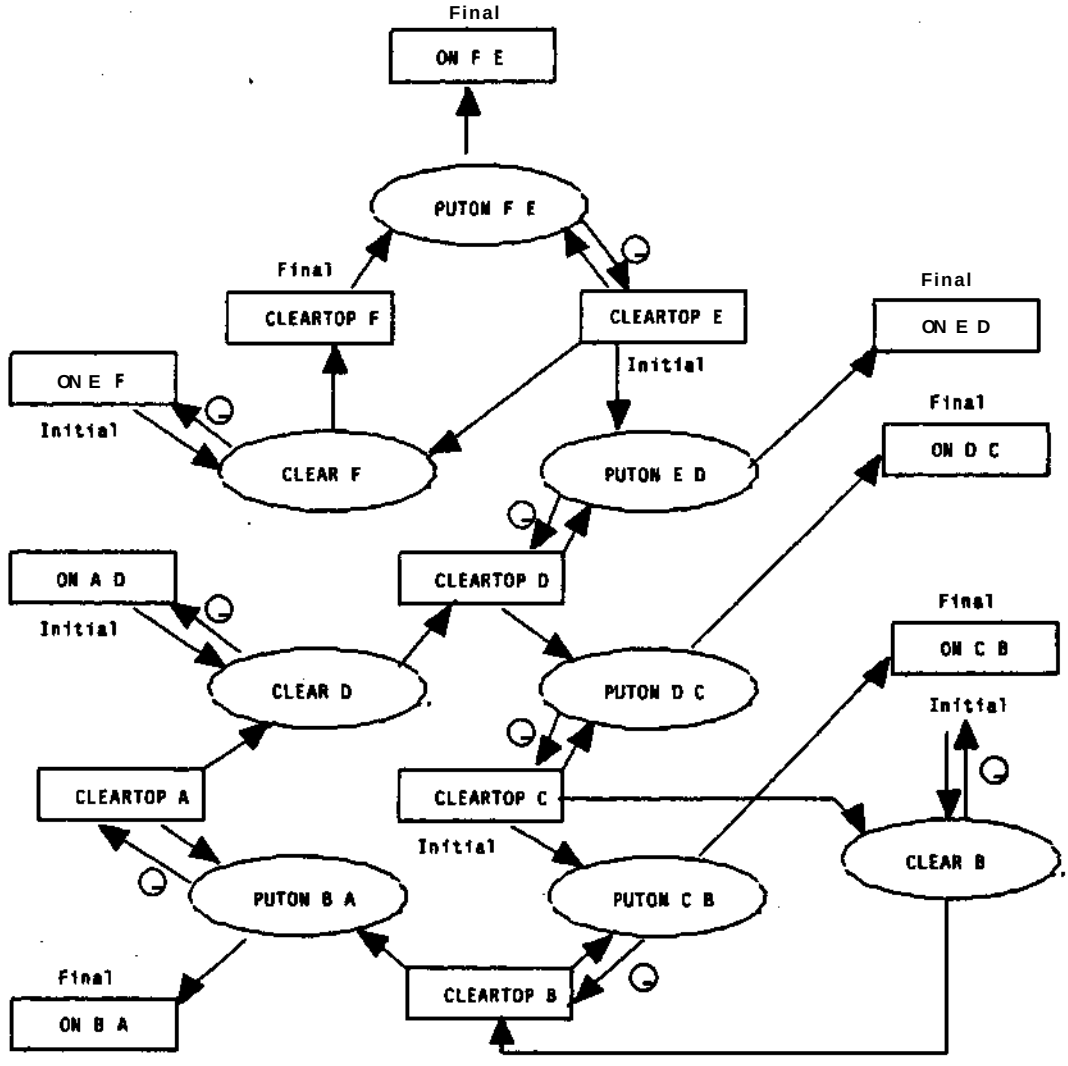


Figure 18: Global Network of Actions and Conditions

An Example with more Actions

The table of abstract PAE's may be modified in order to introduce new possible block movements. One possible action is to move two blocks at the same time and to put **them** on top of a third one. We can call this action *Doubleput,* and define the following **PAE** triplet:

- Action: Doubleput X Y Z
  Preconditions: Block X is on top of block Y; block X is clear on the top **and** block Z is clear on the **top.**
  Effects: Block Z is no longer clear on the top. Block Y is now on top of block Z.

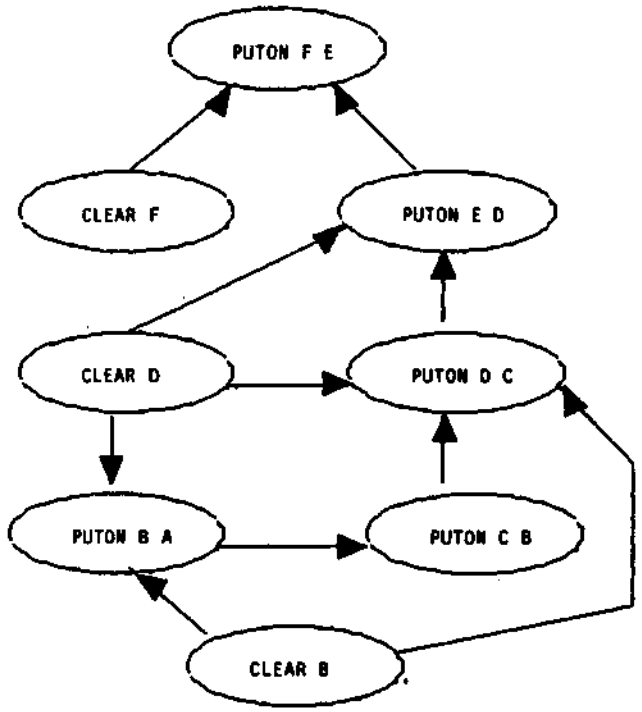


Figure 19: Actions-Network for the Six-Blocks Example

Figure 20 shows a three-blocks problem that was first solved with the original table of PAE's and then with the enlarged one. When only one-block movements are allowed, the resulting plan is:

(Clear A)....> (Clear B)....> (Puton B C)....> (Puton A B)

When two-blocks movements were allowed, the algorithm gave the global network shown in figure 21. Interpretation of tis network leads to the plan with minimum actions:

(Clear A)....> (Doubleput A B C)

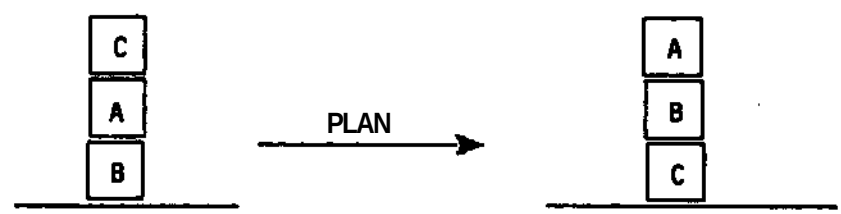


Figure 20: Three-Blocks Example with Two-Blocks Movements

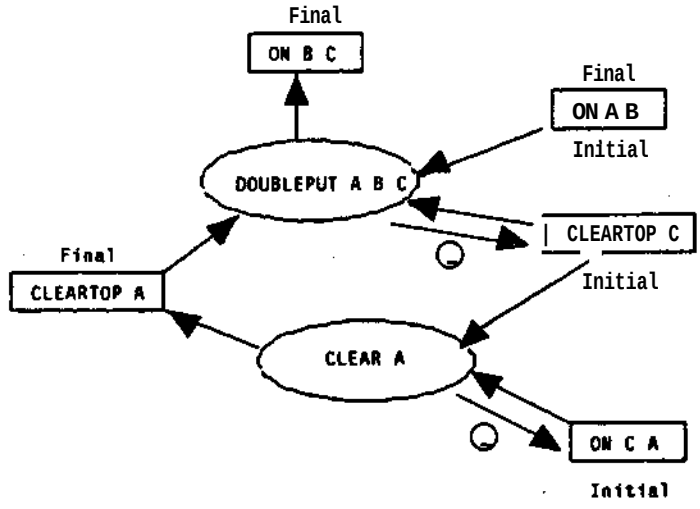


Figure 21: Global Network for the Example with Two-Blocks Movements

Conclusions

Relationships with Other Models

The main differences of the network algorithm with respect to the procedures found in other planning systems such as NOAH and NONLIN are the following:

1. In the network algorithm, both conditions and actions are nodes of the same network. In NOAH and NONLIN, preconditions and effects are part of each action node's body. With this representation a set of critics has to be used to solve the conflicts occurring among the different actions. The network representation does not require of these critics.

2. In the network algorithm, a direct interpretation of the Global Network leads to the Actions Network. Only the relationships among each action and its neighbors have to be identified. This reduces the number of possible relations that have to be studied. In NOAH, the set of possible conflicts that have to be solved is not well defined.

3. The network of actions obtained using the network algorithm can be used to generate several feasible plans. This is different from NOAH's approach where only a first feasible plan is obtained.

It is important to note that the recursive implementation of the condition expansion procedure may not lead to the optimum plan when cost or other action information are incorporated. The only heuristic that was incorporated consisted or ordering the set of Preconditions-Action-Effects triplets with respect to the number of preconditions satisfied. Best-first search or other search strategies could be implemented by labelling

expandable conditions.

Assumptions

The planning algorithm is based on the following assumptions:

1. *Closed Planning Environment:* The system has pre-defined knowledge about all possible operators or actions. The planning problem consist of formulating an appropriate sequence of these operators rather than creating new operators.

2. *Permanent States:* States do not change during time if no operators are applied to the world model.

These assumptions are strong, and it would seem that few problems embody these characteristics. However, our belief is that complex planning problems can be decomposed into subproblems where these assumptions are appropriate. Elaborated planning systems such as OMS [Hayes 85] and MOLGEN [Stefik 81b] also perform on a closed planning environment. MOLGEN has a predefined set of operators structured hierarchically, and OMS has a pre-defined set of *Knowledge Source Activation Records* (KSARs). We do not know about an automated planning system truly capable of creating innovative actions.

Extensions

The presentation of the planning algorithm contains no discussion about the following important aspects of the planning process:

- *Uncertainty of information:* It has been assumed that the outcome of each operator is predictable.

- *Completeness of information:* A full description of the world model was available at any point of the planning process.

- *Heuristics for action choice:* The choice of an action during the backward expansion process was done in terms of feasibility. No considerations about desirability among possible actions were done.

- *Multi-planning Control:* The discussion was centered on the expansion of a single goal (set of states). However, the planning process involves multiple goals.

These aspects are out of the scope of this document. They will be treated in future research, when the network representation and planning algorithm are incorporated into CONSTRUCTION PLANEX, a major planning expert system for construction project

planning [Hendrickson 86].

References

[Corkill 79] **Corkill, D., "Hierarchical Planning in a Distributed Environment,"** *Proceedings of the Sixth International Joint Conference on Artificial Intelligence,* **pp. 168-175,, 1979.**

[Earman 80] **Earman, L.D., Hayes-Roth, F., Lesser, V.R. and Reddy, D.R., 'The** Hearsay-**11 Speech-understanding System: Integrating Knowledge to Resolve Uncertainty,"** *Computing Surveys,* **Vol. 12, pp. 213-253,1980.**

[Hayes 85] **Hayes-Roth, B., "A Blackboard Architecture for Control,"** *Artificial Intelligence,* **Vol. 26, pp. 251-321,1985.**

[Hendrickson 86] **Hendrickson, C.T., Zozaya-Gorostiza C, Rehak, D., Baracco-Miller,** E. **and** P. Lim, *An Expert System Architecture for Construction Planning,* **Technical Report EDRC-12-07-87, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA., 1986.**

[Sacerdoti 74] **Sacerdoti, E.D., "Planning in a Hierarchy of Abstraction Spaces,"** *Artificial Intelligence,* **Vol. 5, No. 2, pp. 115-135,1974.**

[Sacerdoti 77] **Sacerdoti, E.L., *A* *Structure for Plans and Behavior,* Elsevier-Holland, New York, 1977.**

[Sacerdoti 85] **Sacerdoti, E.L., "The Nonlinear Nature of Plans," *Proceedings* of *the*** *Fourth International Joint Conference on Artificial* ***Intelligence,*** **pp. 206-214,1985.**

[Stefik 81 a] **Stefik, M.f "Planning with Constraints (MOLGEN: Part 1),"** *Artificial Intelligence,* **Vol. 16, No.** 2, **pp. 111-140,1981.**

[Stefik81b] **Stefik, M., "Planning and Meta-Planning (MOLGEN: Part** 2)," *Artificial Intelligence,* **Vol. 16, No.** 2, **pp. 141-170,1981.**

[Tate74] **Tate, A.,** *INTERPLAN: A Plan Generation System which can* ***deal with*** *Interactions between Goals,* **Technical Report MIP-R-109, Machine Intelligence Research Unit, University of Edinburgh, 1974.**

[Tate75] **Tate, A.,** *Using Goal Structure to Direct Search in* ***a*** *Problem* ***Solver,*** **unpublished Ph.D. Dissertation, Machine Intelligence Research Unit, University of Edinburgh, 1975.**

[Tate77] **Tate, A., "Generating Project Networks,"** *Proceedings of the Fifth International Joint Conference on Artificial Intelligence,* **pp. 888-893, 1977.**

[Winston 84] **Winston, P.H.,** *Artificial Intelligence,* **Addison-Wesley, Publishing Company Inc., Reading, Mass., 1984.**