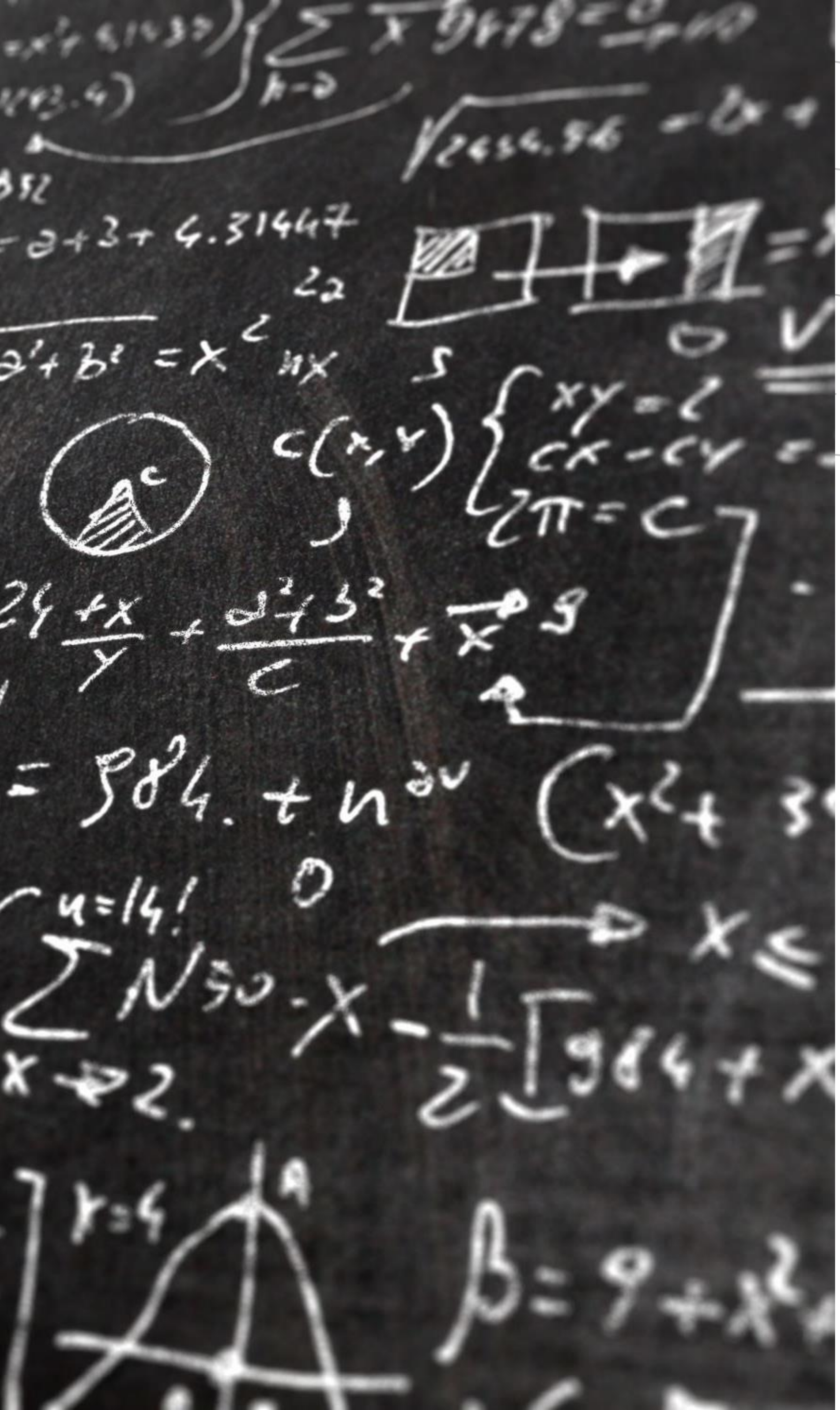


Nonlinear Nature of Plans

Paper by Earl D. Sacerdoti

Presentation by Michael Hentz

Previous Planning

- Planning is linear
  - Finding next step based on previous step and preconditions
- Execution is done in same
  - Needs to be linear
  - Follow plan step by step
- Breadth First Search, STRIPS

```
Given:   L1 = R.(-P⊃Q)
Obtain:  L0 = (QvP).R

Goal 1:  Transform L1 into L0.
   Match produces position difference (ΔP).
   Goal 2:  Reduce ΔP between L1 and L0.
      First operator found is R1.
      Goal 3:  Apply R1 to L1.
         Goal 4: Transform L1 into C(R1).
            Match succeeds with A = R and B = -P⊃Q.
         Produce new object:

            L2 = (-P⊃Q).R

   Goal 5:  Transform L2 into L0.
      Match produces connective difference (ΔC) in left subexpression
      Goal 6:  Reduce ΔC  between left of L2 and left of L0.
         First operator found is R5.
         Goal 7:  Apply R5 to left of L2.
            Goal 8:  Transform left of L2 into C(R5).
               Match produces connective difference (ΔC) in left subexpression.
               Goal 9:  Reduce ΔC between left of L2 and C(R5).
                  Goal rejected: difference is no easier than difference in
                  Goal 6.
         Second operator found is R6.
         Goal 10:  Apply R6 to left of L2.
            Goal 11:  Transform left of L2 into C(R6).
               Match succeeds with A = -P and B = Q.
            Produce new object:

               L3 = (PvQ).R

   Goal 12:  Transform L3 into L0.
      Match produces position difference (ΔP) in left subexpression.
      Goal 13:  Reduce ΔP between left of L3 and left of L0.
         First operator found is R1.
         Goal 14:  Apply R1 to left of L3.
            Goal 15:  Transform left of L3 into C(R1).
               Match succeeds with A = P and B = Q.
            Produce new object:

               L4 = (QvP).R

      Goal 16:  Transform L4 into L0.
         Match shows L4 is identical with L0, QED.
```

Planning Does Not have to be **Linear**

Can separate **planning** from **execution**

Total Planning vs Partial Planning

TOTAL PLANNING

- Strict ordering of actions with respect to time
- Planning that involves solving each subgoal at a time
- Linear Planning
- Step-by-step

PARTIAL PLANNING

- Partial ordering of actions with respect to time
- Planning that involves solving each subgoal independently, and then finding interactions
- Semi-linear Planning
- Subgoals reached

Partial Planning Overview

- Treat each subgoal as separate
- Solve each independently
- Figure out how subgoal solutions interact and rephrase
- Repeat

Why Partial Planning

Freedom of Ordering in Planning

- No backtracking or wasted computation
- No random choice based on missing information
  - Ex: Move to multiple spots

Deal with more manageable information

- Smaller tests that studying full plan

Certain Problems can be optimally solved

- Unlike STRIPS

Partial Planning Shortcomings

- Disjunctive Goals
- Nonlinearizable Interactions
  - Achieving subgoals cannot achieve goal
- Disjunctive Outcomes of Actions
  - When outcomes change possible outcomes of other actions
  - If…else…

How to Implement Partial Planning

Procedural Net

Linked subgoal nodes

- Simulating nodes create more nodes and paths
- Paths are plans

Each node contains:

- Code
- Predecessor step(s)
- Successor step(s)
- Add list
- Delete list

Types of nodes:

- Goal – Goal to be achieved
- Phantom – Goals that should be true at point in path
  - Precondition goal
- Split – Split Path
- Join – Join Paths
  - Both Paths must occur before moving beyond Join

Planning Algorithm

NOAH

1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan with reordering and elimination
4. Repeat at step 2 until no further plan can be made
   - No New Details of subgoals

Critics

- Resolve Conflict
  - Preconditions of one subplan path affected by another subplan path
  - Reorder and redirect paths
- Use Existing Objects
  - For steps with multiple objects, will not bind object unless fully optimal
  - Will create "formal object" as placeholder
- Eliminate Redundant Preconditions
  - Same nodes in subplan paths
  - Delete same node on all but one path

Example

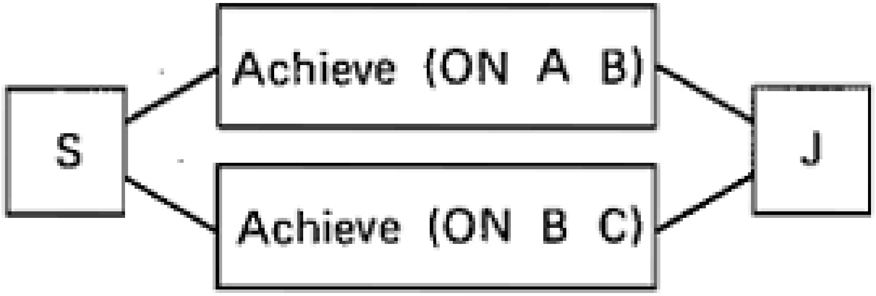
1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan
    1. Resolve Conflict
    2. Use Existing Objects
    3. Eliminate Redundant Preconditions
4. Repeat at step 2 until no further plan can be made

Start With Goal

Achieve (AND(ON A B)(ON B C))

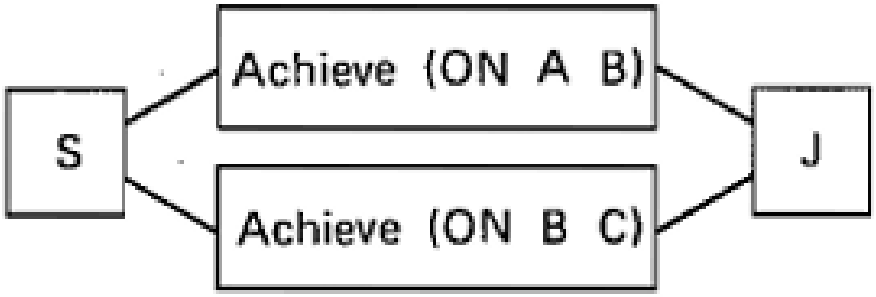
1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan
   1. Resolve Conflict
   2. Use Existing Objects
   3. Eliminate Redundant Preconditions
4. Repeat at step 2 until no further plan can be made

Break Down Conjunctive Goal into subgoals



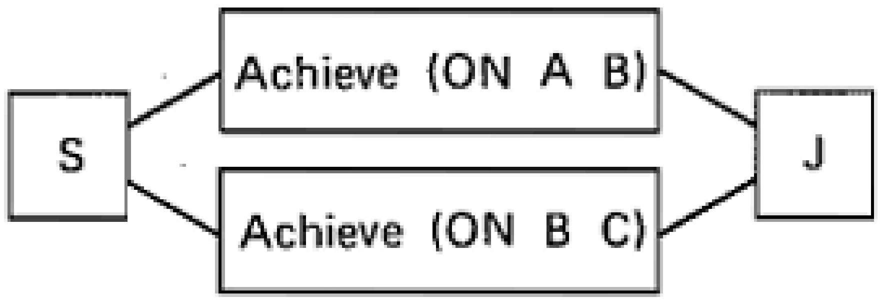
1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan
    1. Resolve Conflict
    2. Use Existing Objects
    3. Eliminate Redundant Preconditions
4. Repeat at step 2 until no further plan can be made

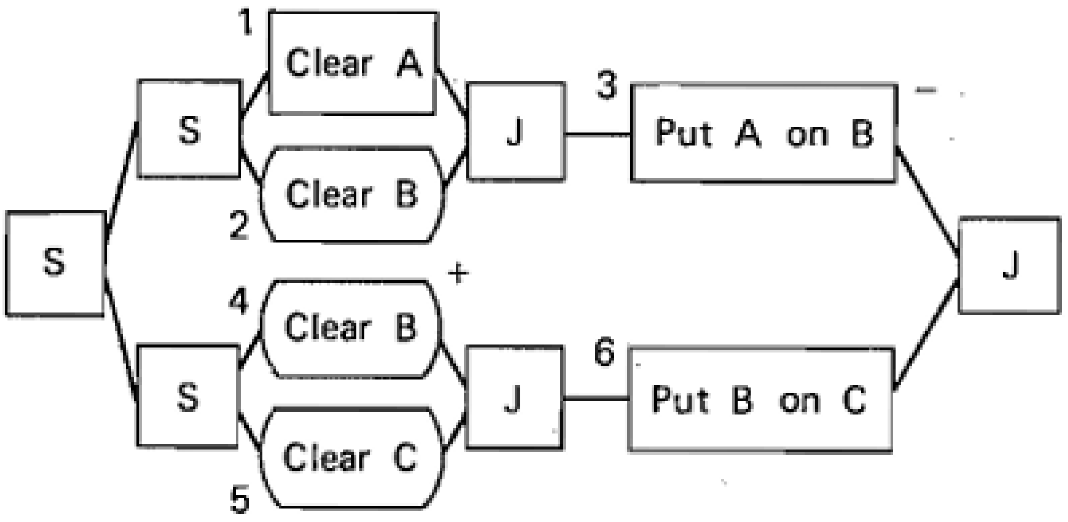
No problems found from critics



1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan
    1. Resolve Conflict
    2. Use Existing Objects
    3. Eliminate Redundant Preconditions
4. Repeat at step 2 until no further plan can be made

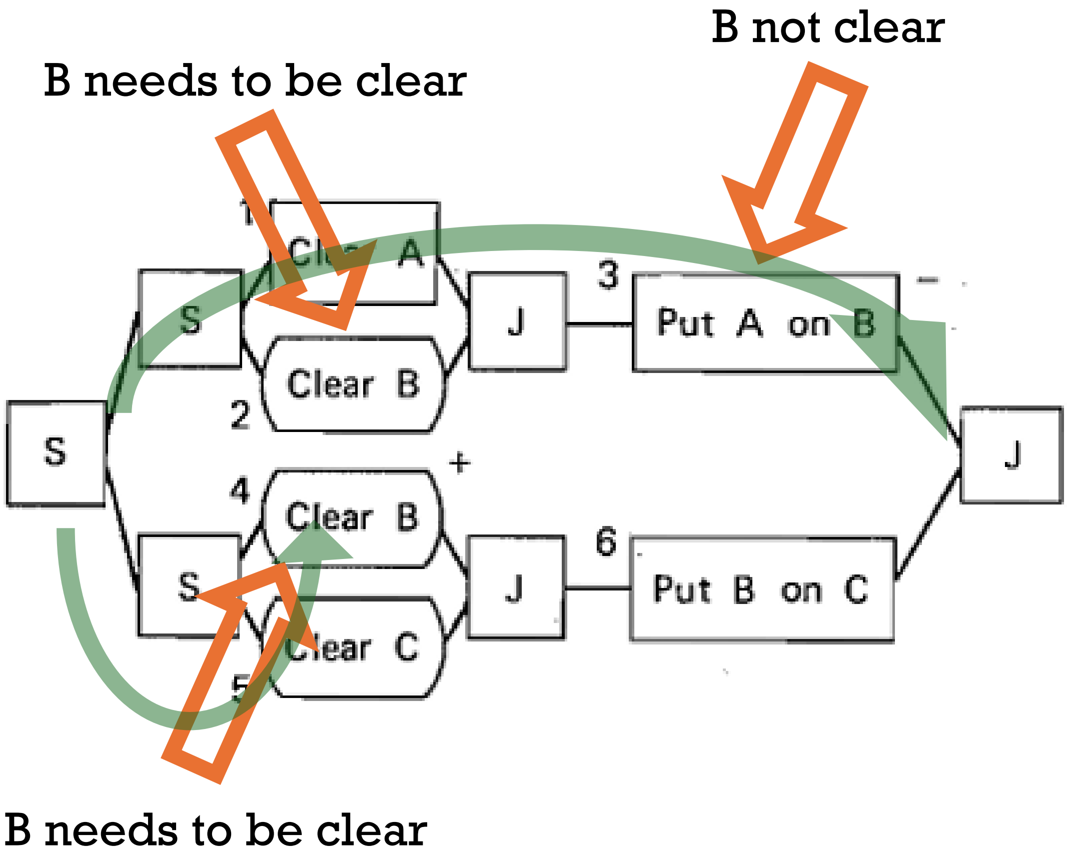
- For (ON X Y), need subgoal of Put X on Y
- For Put X on Y, need Clear X and Clear Y
    - Can be broken down





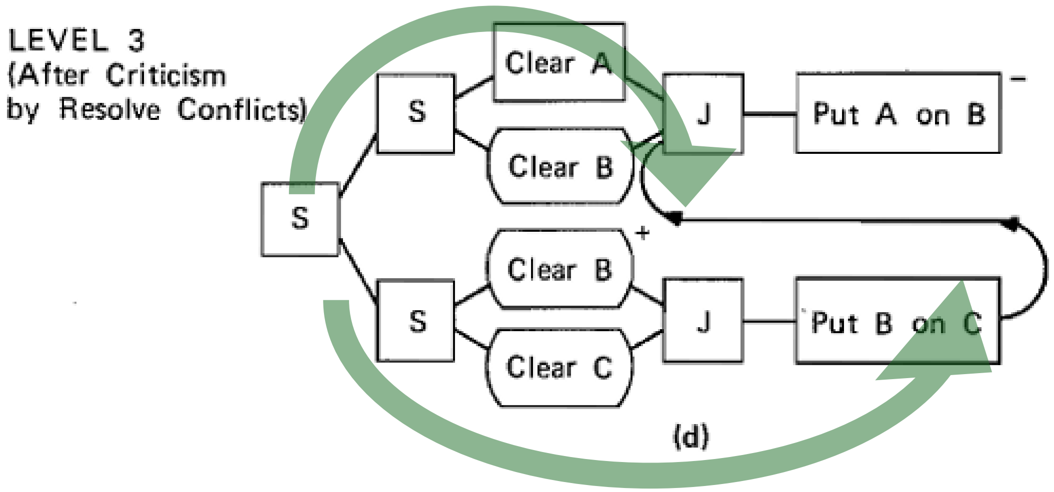
1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan
    1. Resolve Conflict
    2. Use Existing Objects
    3. Eliminate Redundant Preconditions
4. Repeat at step 2 until no further plan can be made

- Use list of adds and deletes to find precondition conflict
- Ignore if precondition is denied by goal



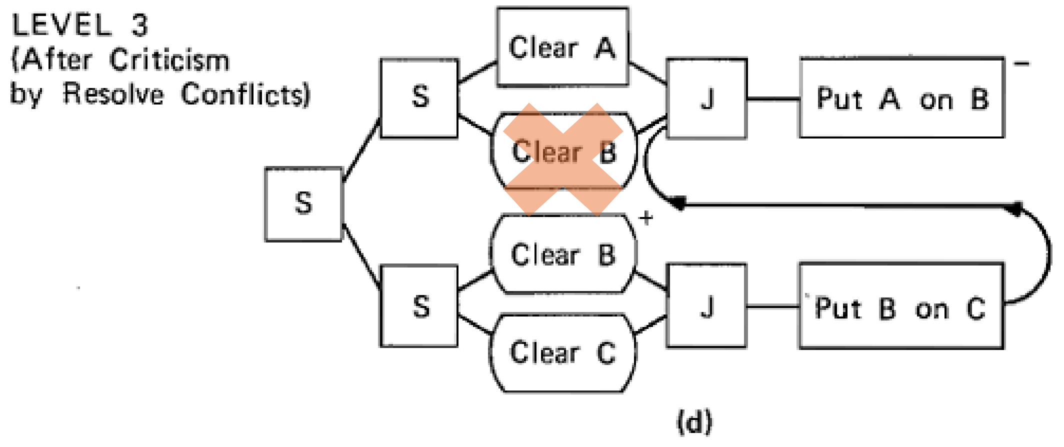
1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan
    1. Resolve Conflict
    2. Use Existing Objects
    3. Eliminate Redundant Preconditions
4. Repeat at step 2 until no further plan can be made

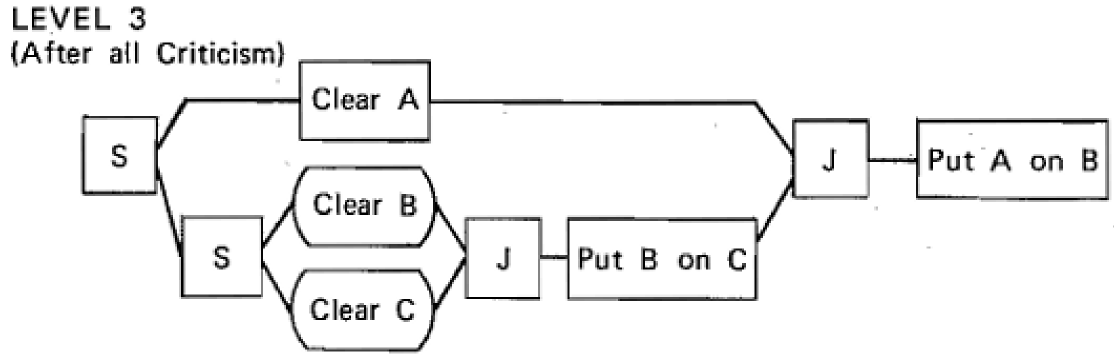
- Reorder subplan paths so subgoal is before step that violates subgoal



1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan
    1. Resolve Conflict
    2. Use Existing Objects
    3. Eliminate Redundant Preconditions
4. Repeat at step 2 until no further plan can be made

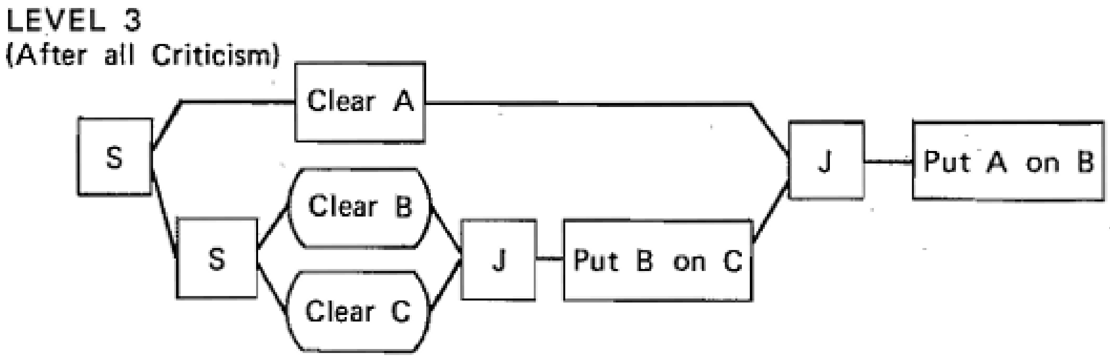
- Removed redundant Clear B
    - Either redundant deletion is viable

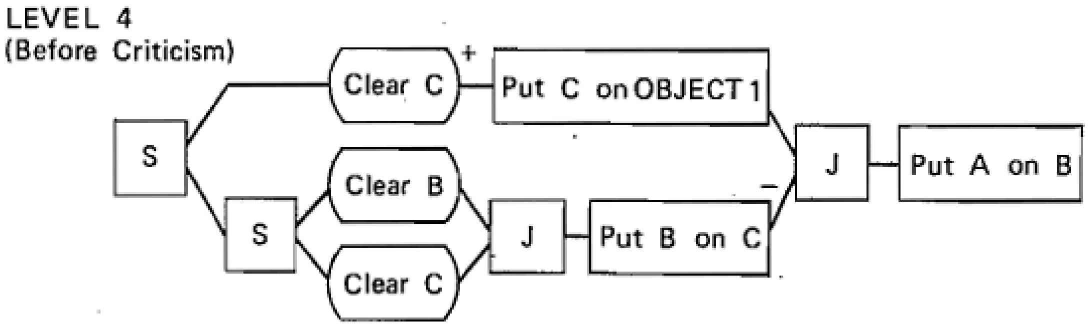




1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan
    1. Resolve Conflict
    2. Use Existing Objects
    3. Eliminate Redundant Preconditions
4. Repeat at step 2 until no further plan can be made

- Move C so A is clear
- Do not yet state where C is moved to
    - Could be Table or could be another available block

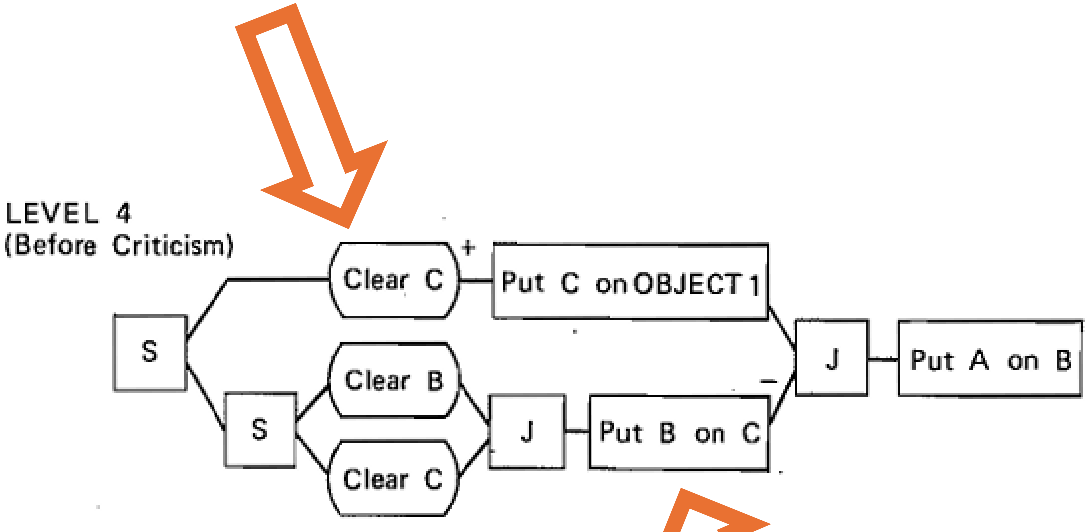




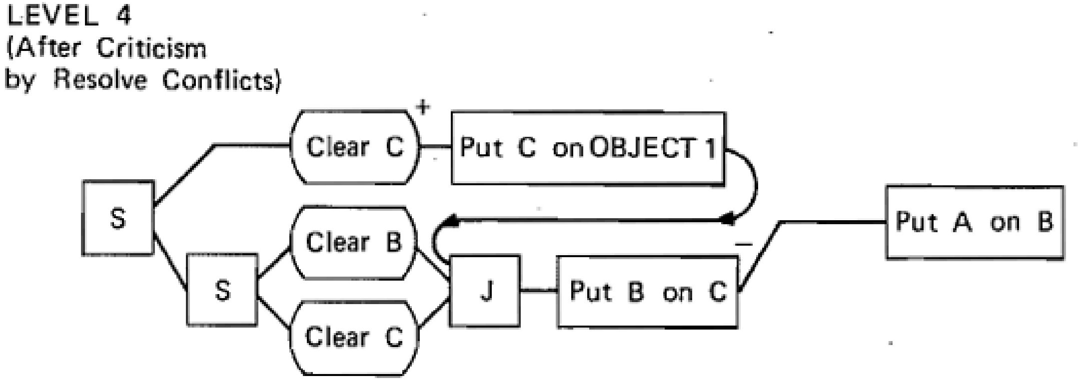
(f)

1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan
    1. Resolve Conflict
    2. Use Existing Objects
    3. Eliminate Redundant Preconditions
4. Repeat at step 2 until no further plan can be made

- If B is on C, then C cannot be clear and cannot be moved



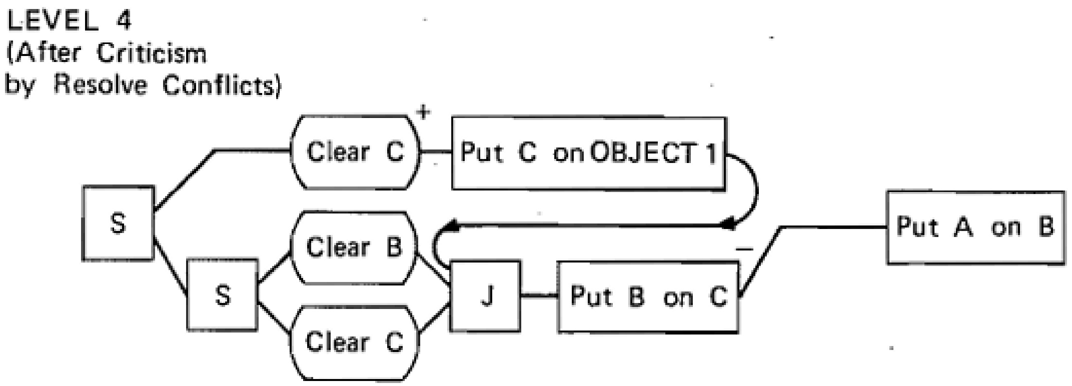
(f)



(g)

1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan
    1. Resolve Conflict
    2. Use Existing Objects
    3. Eliminate Redundant Preconditions
4. Repeat at step 2 until no further plan can be made

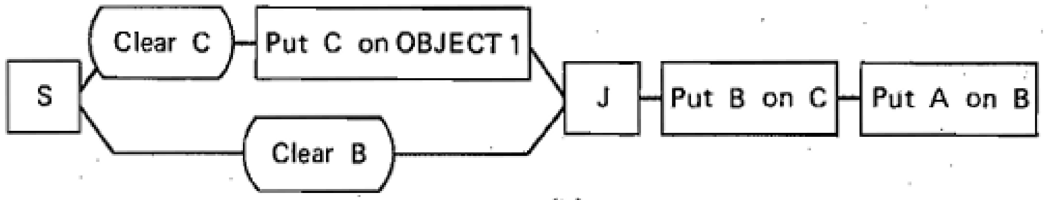
- Eliminate Clear C



(g)

1. Start with Goal
2. Break down goal into basic, most detailed steps by simulating subgoal nodes
3. Criticize new plan
    1. Resolve Conflict
    2. Use Existing Objects
    3. Eliminate Redundant Preconditions
4. Repeat at step 2 until no further plan can be made

All subgoals are satisfied



(h)

Partial Planning

- Separate total execution from partial planning
- As opposed to the strict ordering of total planning, partial planning is linear and has more freedom of order
  - Less backtracking and waste
  - Less random unoptimized choices
- Planning:
  - Treat each subgoal as separate
  - Solve each independently
  - Figure out how subgoal solutions interact and rephrase
  - Repeat
- Does not work with Disjunctions and nonlinearizable goals