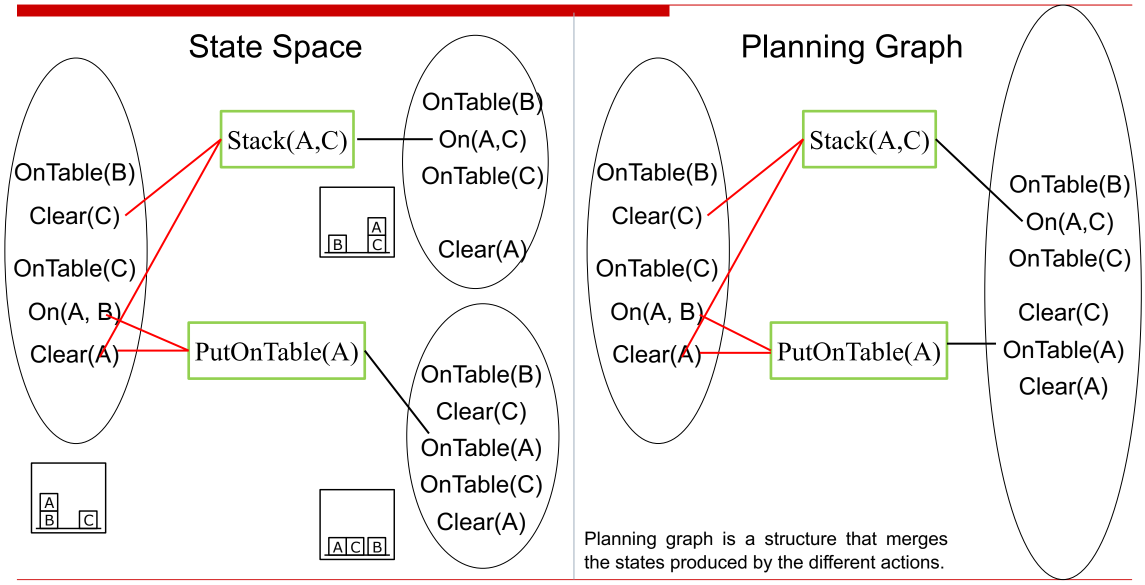
Fundamentals of Artificial Intelligence

Planning Graphs and GraphPlan

Shyamanta M Hazarika
Mechanical Engineering
Indian Institute of Technology Guwahati
s.m.hazarika@iitg.ac.in

http://www.iitg.ac.in/s.m.hazarika/

State Space and Planning Graph



Planning graph is a structure that merges the states produced by the different actions.

State Space and Planning Graph

- Basic difference between the state space for a planning problem and the corresponding planning graph
  - State space applies an action to a given state and generates a successor state.
  - For each action that is applicable, it generates a separate candidate state, which is a starting point for further exploration.
  - Planning Graph merges the states produced by the different actions that are applicable.
  - Resulting set of propositions expressed as a single layer; as also the actions that resulted in this layer.

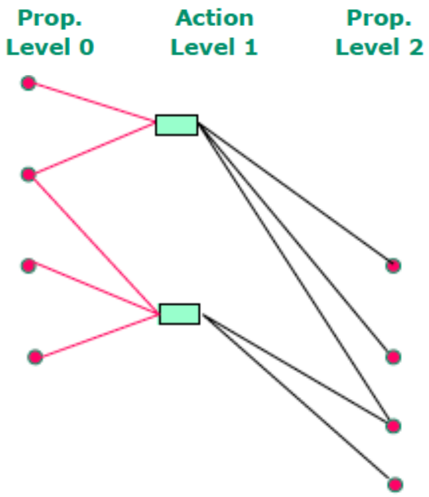
3

Planning Graph

- A layered graph
  - Two kinds of layers alternate
    - Literal (propositions)
    - Action
- The first layer is a literal layer which shows all the literals that are true in the initial layer
  - Every action has a link from each of its preconditions and a link to each of its effects.
  - Literals at consecutive literal levels linked through NoOps
- Every two layers corresponds to a discrete time
  - Propositions (in a layer) that cannot be true simultaneously; and actions that cannot happen simultaneously define a mutex.

4

Planning Graph



A plan graph is LAYERED. Have a bunch of alternating layers of propositions and actions. Start with level zero, level one, level two.

At the even- numbered levels you have propositions (Here drawn as a little dot). At the odd-numbered levels, you have actions, shown as boxes.

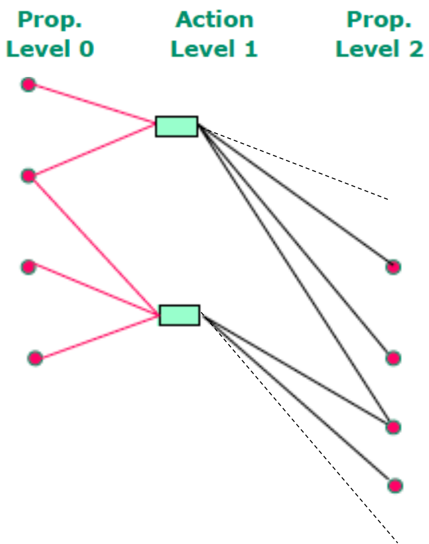
There are FOUR kinds of EDGES:

Precondition Edge
Add to layer 1 all the actions that have their preconditions satisfied in layer 0.

Positive Edge
Add to proposition layer 2 all the propositions that are effects of actions in layer 1.

5

Planning Graph



A plan graph is LAYERED. Have a bunch of alternating layers of propositions and actions. Start with level zero, level one, level two.

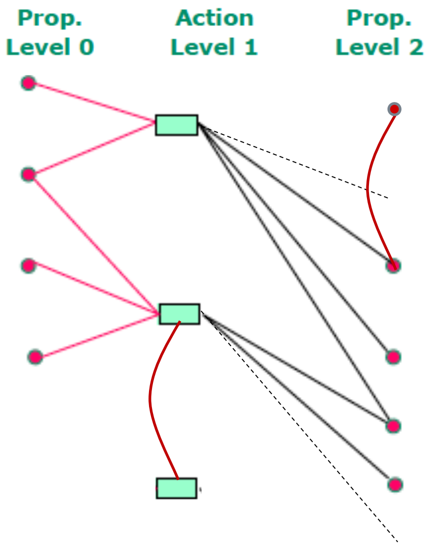
At the even- numbered levels you have propositions (Here drawn as a little dot). At the odd-numbered levels, you have actions, shown as boxes.

There are FOUR kinds of EDGES:

Negative Edge
Linking an action to delete effect of the action; propositions that no longer true.

6

Planning Graph



A plan graph is LAYERED. Have a bunch of alternating layers of propositions and actions. Start with level zero, level one, level two.

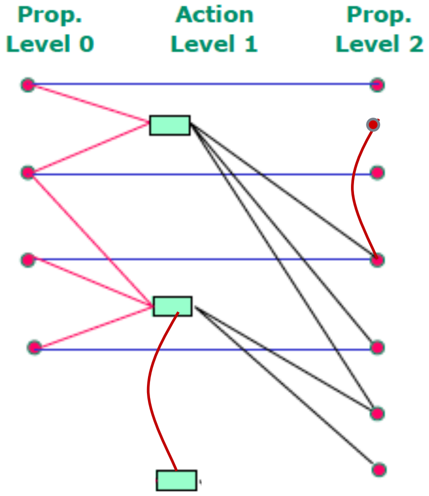
At the even- numbered levels you have propositions (Here drawn as a little dot). At the odd-numbered levels, you have actions, shown as boxes.

There are FOUR kinds of EDGES:

Negative Edge
Linking an action to delete effect of the action; propositions that no longer true.

Mutex Edge
Linking mutually exclusive two propositions in same layer; or two actions in same layer.

Planning Graph

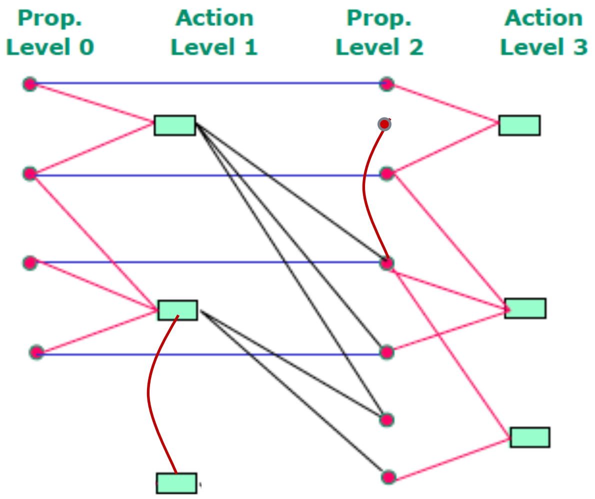


Add to next proposition layer, all of the propositions that you had in the previous layer. **Connect them with maintenance actions or No-ops**, shown here as blue lines connecting propositions from layer n to layer n + 2.

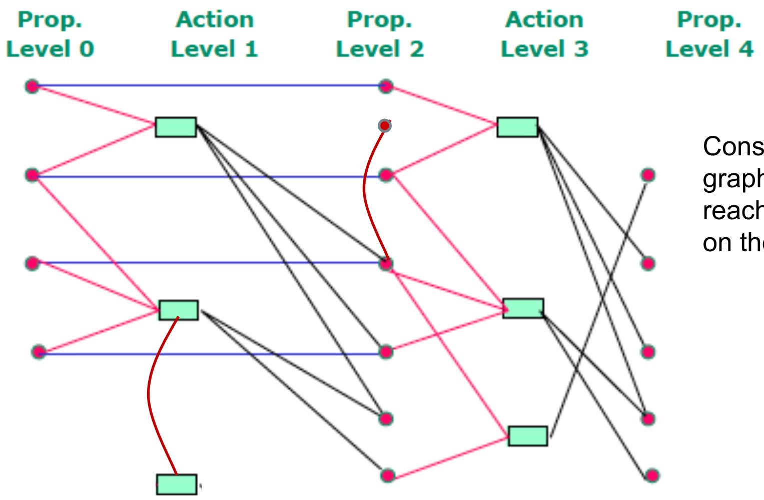
These maintenance actions represent the possibility of having some proposition be true at step n because it was true at step n – 2, and we didn't do anything to make it false; that is, that we maintained its truth value.

The effect of the no-ops is that the set of propositions in the proposition layers grow monotonically.

Planning Graph

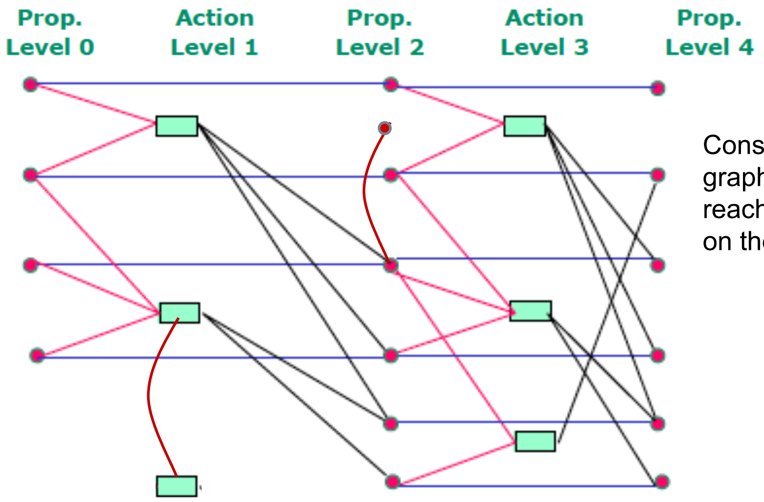


Planning Graph



Construction of the planning graph is some kind of reachability analysis done on the given problem.

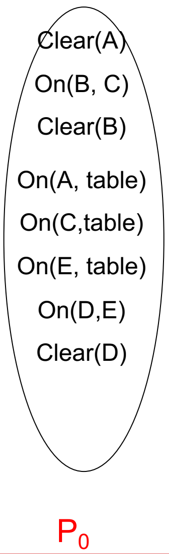
Planning Graph



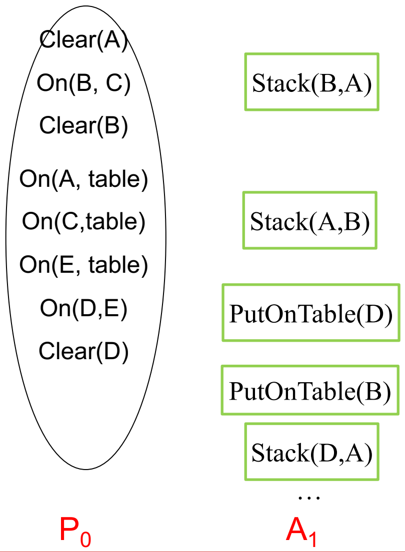
Construction of the planning graph is some kind of reachability analysis done on the given problem.

We have **HERE three proposition levels** (levels 0, 2, and 4) and **two action levels** (levels 1 and 3). **Encode depth-two plans** (There are 02 layers of actions). Action at Level 1 - actions to do first step; Action at level 3 - actions to do on the second step.

Illustrative Example



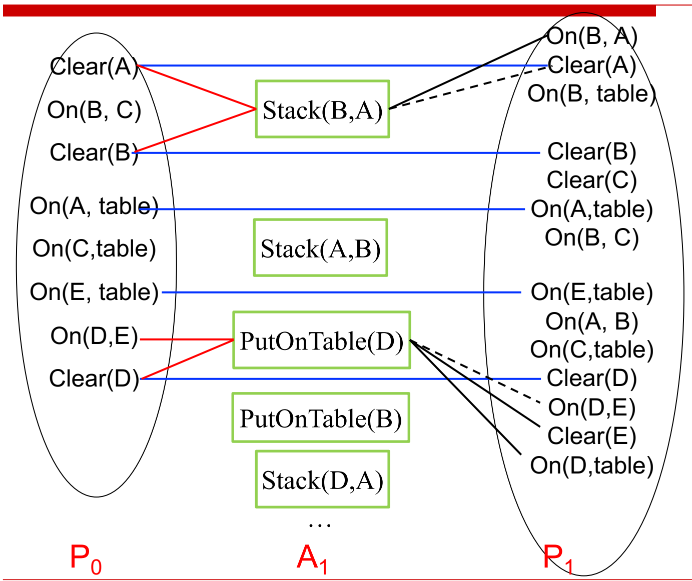
Illustrative Example



13

© Shyamanta M Hazarika, ME, IIT Guwahati

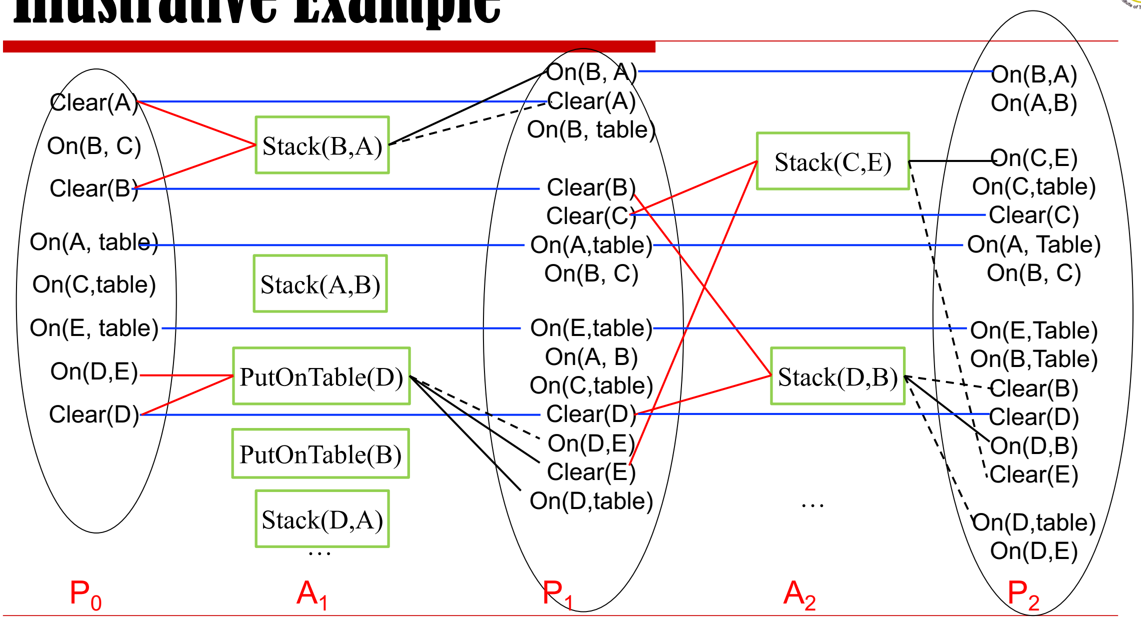
Illustrative Example



14

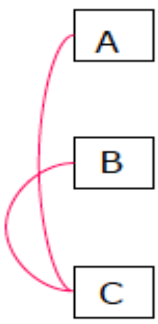
© Shyamanta M Hazarika, ME, IIT Guwahati

Illustrative Example



Mutually Exclusive Actions

Find and mark pairs of actions that are mutually exclusive.



That is, they can't both be done on the same step; they can't be done in parallel. When two actions are mutex, we'll show it in our graph by drawing an arc between them, as shown in red

Here. Actions A and C are mutex, and actions B and C are mutex.

This means that we could execute both A and B in parallel, but if we do C, we can't do either of the others.

Mutually Exclusive Actions

1. Inconsistent effects: effect of one action is negation of effect of another. The semantics of these actions in parallel are not defined.

Mutually Exclusive Actions

1. Inconsistent effects: effect of one action is negation of effect of another The semantics of these actions in parallel are not defined.

2. Interference: one action deletes the precondition of the other; Only one linear ordering of the two actions would be feasible.

Mutually Exclusive Actions

1. Inconsistent effects: effect of one action is negation of effect of another

2. Interference: one action deletes the precondition of the other; Only one linear ordering of the two actions would be feasible.

3. Competing needs: actions have preconditions that are mutex at level i-1.

Mutually Exclusive Propositions

Two propositions at level i are mutex

1. Negation: they are negations of one another

Mutually Exclusive Propositions

Two propositions at level i are mutex

1. Negation: they are negations of one another
2. Inconsistent support: all ways of achieving the propositions at level i-1 are pairwise mutex.

Observe that if two propositions are not mutex in a layer, they will be not mutex in all subsequent layers because of the No-op action.

GraphPlan

- A propositional planner

The idea is that you make a plan graph of depth k, and then you search for a solution, and if you succeed you return a plan. Otherwise, you increment K and try again.

1. Make a plan graph of depth k
2. Search for a solution
3. If succeed, return a plan
4. Else k=k+1
5. Go to 1

The process of building the planning graph continues till either the goal propositions have appeared nonmutex in the planning graph or the planning graph has levelled off.

GraphPlan

- Phase 1 – Planning Graph Construction
  - Creates graph encoding pairwise consistency and reachability of actions and propositions from initial state.
  - Graph includes, as a subset, all plans that are complete and consistent.
- Phase 2 - Solution Extraction
  - Graph treated as a kind of constraint satisfaction problem (CSP).
  - Selects whether or not to perform each action at each time point, and testing consistency.

Building the Planning Graph

- First task that GraphPlan takes up is construction of the planning graph.
- Initialize the layer zero of the planning graph to the start state!
- Process of extending the graph continues till one of the following two conditions is achieved.
  - Newest proposition layer contains all goal propositions; and no mutex relations between the goal propositions.
  - The planning graph has levelled off. This means that for two consecutive levels
    - Same set of Propositions; Same set of mutex relations
    - Consequently no new actions can make an appearance; If goal propositions are not present in a levelled planning graph, they can never appear!

Solution Extraction

- When the planning graph reaches a level that contains all the goal propositions, it is time to check if a solution can be extracted.

1. One way of looking at the plan existence question is to see if the goal propositions have support from a nonmutex set of actions.
2. If YES! combined preconditions can be seen as a regressed sub-goal set, which can be solved recursively.

- If all the literals in the goal appear at the deepest level and non mutex, then search for a solution for each subgoal at level i

1. For each subgoal at level i
   a. Choose an action to achieve it
   b. If it's mutex with another action, Fail

25

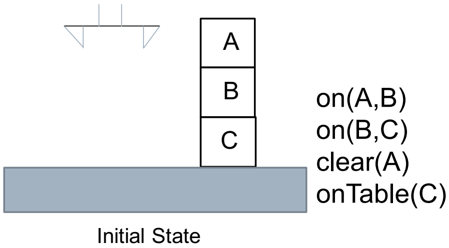
Solution Extraction

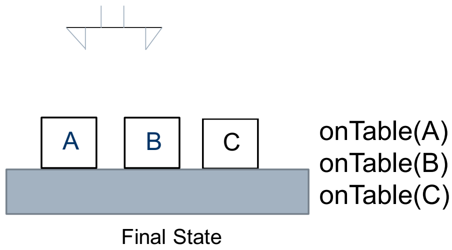
- When the planning graph reaches a level that contains all the goal propositions, it is time to check if a solution can be extracted.
- If all the literals in the goal appear at the deepest level and non mutex, then search for a solution for each subgoal at level i

1. For each subgoal at level i
   a. Choose an action to achieve it
   b. If it's mutex with another action, Fail.
2. Repeat for preconditions at level i-2

26

Illustrative Example

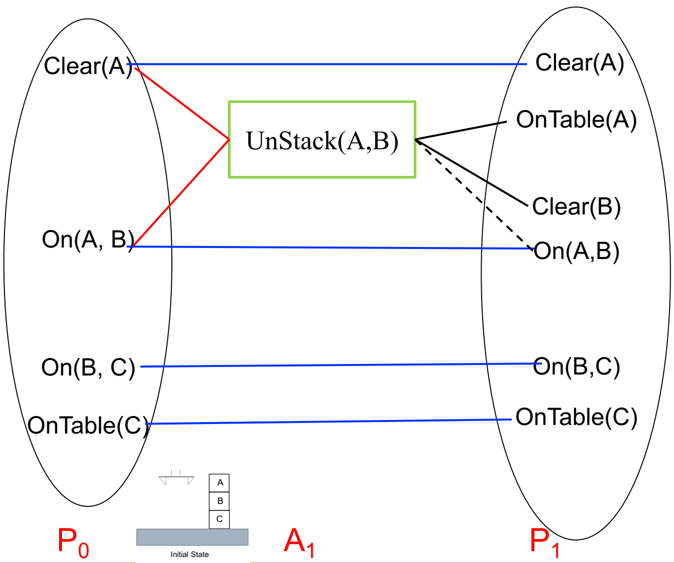




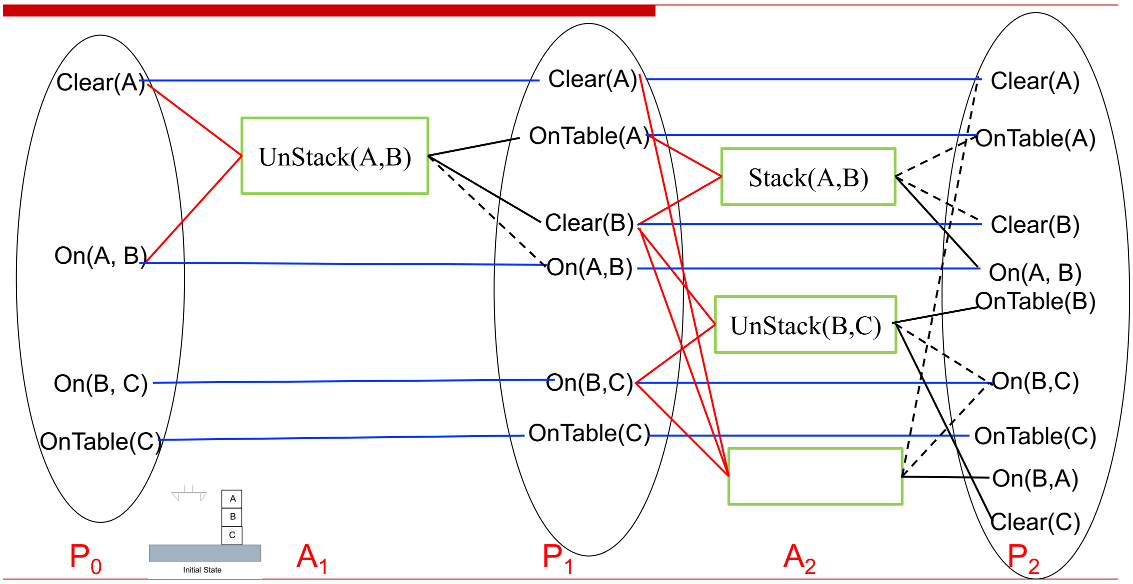
Stack(x,y)
PRE: {OnTable(x),Clear(x), Clear(y)}
ADD: {On(x,y)}
DEL: {OnTable(x),Clear(y)}

UnStack(x,y)
PRE: {On(x,y),Clear(x)}
ADD: {OnTable(x), Clear(y) }
DEL: {On(x,y)}

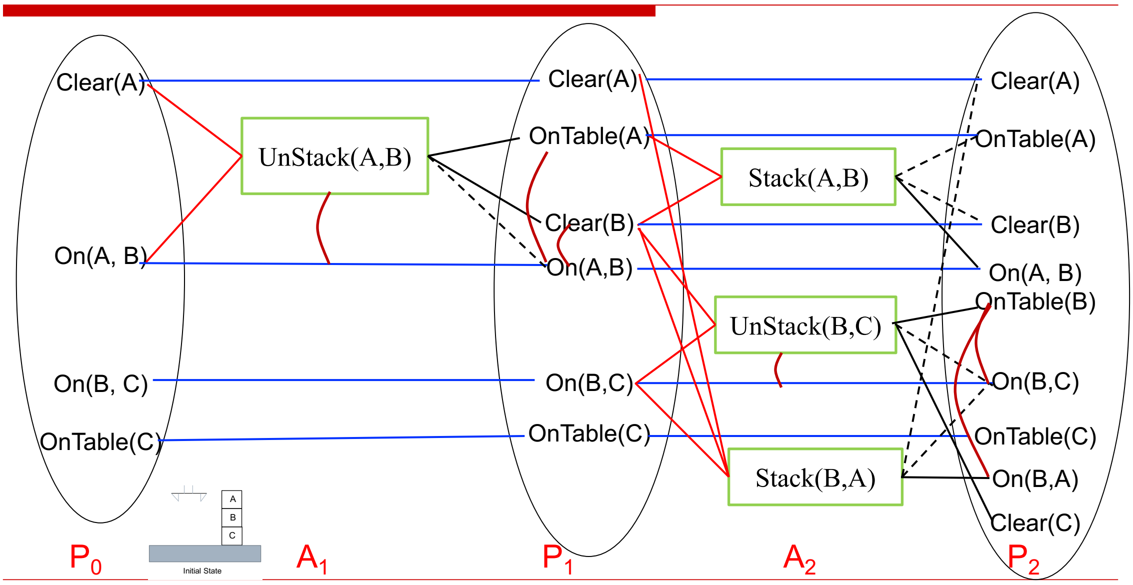
Illustrative Example



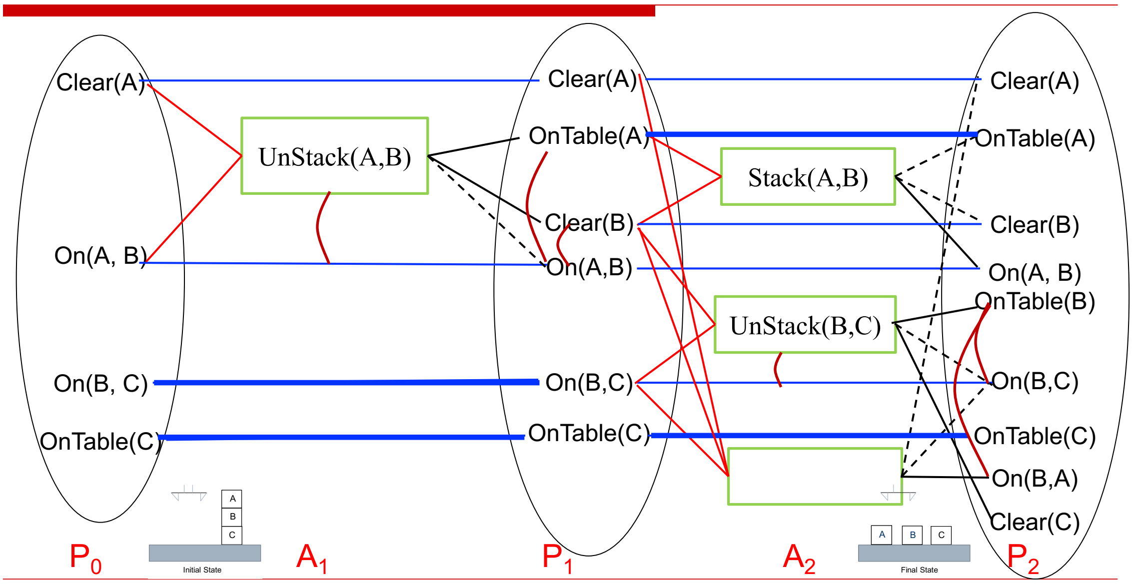
Illustrative Example



Illustrative Example

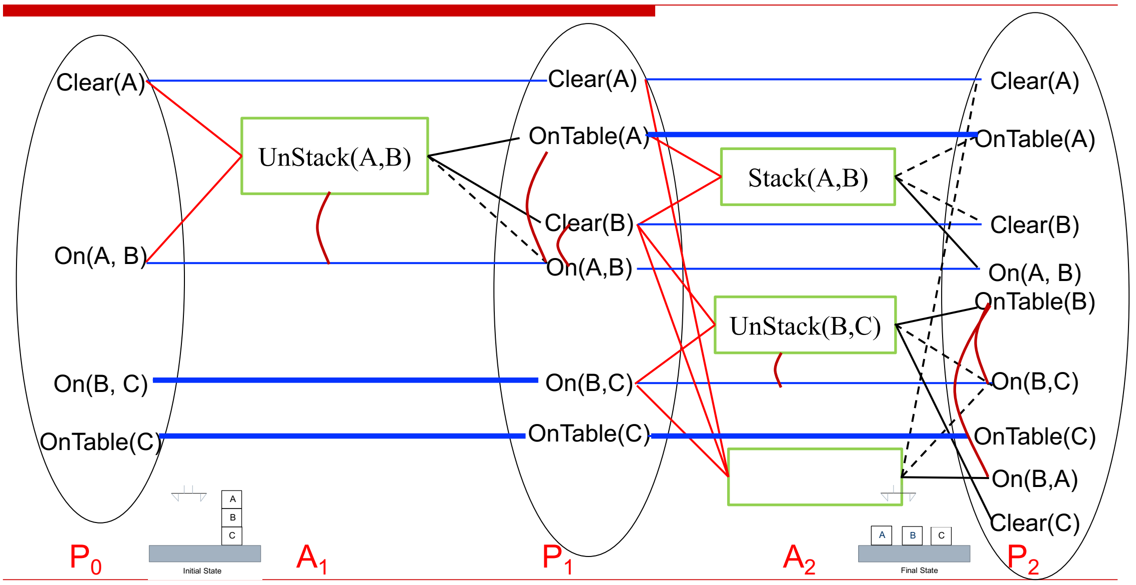


Illustrative Example



31

Illustrative Example



32

Final Comments

- Development of GraphPlan gave a huge impetus to the research in planning.
  - Graphplan – introduced the two stage process to planning.
  - First Stage involved delimiting the search space; and some kind of reachability analysis done on the given problem.
  - Second Stage was searching for a plan in a delimited space.
- Advancements in planning algorithms and increasing computing power led to the exploration of richer domains for planning.
  - Conditional effects, contingent planning; planning with durative actions and trajectory constraints have been explored.
  - Planning techniques applied to many real world applications!