G52PAS Revision of goal stack planning, partial order planning, and GraphPlan

Consider the following planning problem (for the sake of GraphPlan the actions are already propositionalised):

Objects: $Home, Supermarket, Milk, Bananas$

Fluents: $At(Home), At(Shop), Have(Milk), Have(Bananas)$

Actions:

$Go(Home)$:
PRECOND: $At(Shop)$
EFFECT: $At(Home), \neg At(Shop)$

$Go(Shop)$:
PRECOND: $At(Home)$
EFFECT: $At(Shop), \neg At(Home)$

$Buy(Milk)$:
PRECOND: $At(Shop)$
EFFECT: $Have(Milk)$

$Buy(Bananas)$:
PRECOND: $At(Shop)$
EFFECT: $Have(Bananas)$

Initial state: $At(Home)$

Goal: $At(Home), Have(Bananas), Have(Milk)$

1. Solve it with goal stack planning algorithm
2. Solve it with partial order planning algorithm
3. Solve it with GraphPlan

Goal stack Uses: a stack, a knowledge base KB, and a plan (a list of actions). To start with, the stack has the goal state, the plan is empty, and KB contains the initial state. Repeat until the stack is empty:

1. If stack top is a compound goal, push its unsatisfied (with respect to the KB) subgoals on the stack.
2. If stack top is a single unsatisfied goal, replace it by an action that makes it satisfied and push the action's precondition on the stack.
3. If stack top is an action, pop it from the stack, execute it and change the knowledge base by the action's effects, and add it to the plan
4. If stack top is a satisfied goal, pop it from the stack.

Partial Order Planner Uses: a set of steps (actions), a set of conditions (fluents), a list of causal links of the form $Action1 \overset{Condition\ C}{\longrightarrow} Action2$ (action 1 is performed to make a precondition C of action 2 true), and a list of temporal constaints of the form $Action1 \prec Action2$. An open condition is a condition that has no causal link making it true.

Starts with a set of two steps $Start$ and $Finish$ such that $Start \prec Finish$ and $Start$ has as its effects the initial state description and $Finish$ has as its preconditions the goal state description. Repeat until there are no open conditions:

1. Choose a step S_{need} which has an open condition C. Find a step S_{add} either in the existing steps or in the list of actions which makes C true. If S_{add} is not in the existing steps, add it to the set of steps.
2. Add a link $S_{add} \xrightarrow{C} S_{need}$, and temporal constraint $S_{add} \prec S_{need}$. If S_{add} is new, add also $Start \prec S_{add}$ and $S_{add} \prec Finish$
3. Check for clobbering: for all causal links $Action1 \overset{Condition\ C}{\longrightarrow} Action2$ check if there is $Action3$ which makes C false. If yes add either $Action3 \prec Action1$ or $Action2 \prec Action3$ to temporal constraints.

GraphPlan Uses: a planning graph which consists of alternating layers S_i and A_i where S_i is a set of fluents (positive and negative) and A_i a set of actions whose preconditions are in S_i, including no-op actions which just preserve a fluent. The next level S_{i+1} has effects (positive and negative) of executing actions in A_i.

Mutex links added between two actions at the same level if any of the following three conditions holds:

1. Inconsistent effects: one action negates an effect of another.
2. Interference: one of the effects of one action is the negation of a precondition of the other.
3. Competing needs: one of the preconditions of one action is mutually exclusive with a precondition of the other.

Mutex holds between fluents if:

1. they are negations of each other
2. each possible pair of actions that could achieve the two literals is mutually exclusive

To construct a planning graph, start with S_0 that contains fluents which are true in the initial state (and negations of those which are false) and keep going (adding levels and mutex links) until no change.

To extract a plan, start in the last level and do backward search:

- The initial state is the last level of the planning graph, S_n, along with the set of goals
- Available actions in S_i: any set of conflict-free actions in A_{i-1} whose effects cover the goals in the state. Conflict-free means: no two actions are mutex and no two of their preconditions are mutex
- The result of applying it is a subset of S_{i-1} which has as its set of goals the preconditions of the selected set of actions.
- The goal is to reach a state at level S_0 such that all the goals are satisfied.