Planning

Planning

Planning Definitions

- *Planning* is finding and choosing a sequence (or a "program") of actions to achieve goals.
- A planning *state* is described by specifying which *positive literals* are true.
- The *goal* is described by specifying which literals should be true and false.
- *Actions* are described by specifying what changes occur.
- *Search* can perform planning. Planning states map to search states, actions to operators.
- A *progression* planner searches from the iniitial state to the goal. A *regression* planner searches from the goal to the initial state.

Planning States and Goals

States and goals can be represented by conjunctions of positive literals.

- Example: Blocks-world with a table T and three blocks named A, B, and C.
- Positive Literals:
  $on(A,T), on(A,B), on(A,C), \ldots$
  $clear(A), clear(B), clear(C)$
- The state where C is on A and B is by itself:
  $clear(C) \wedge on(C,A) \wedge on(A,T)$
  $\wedge\, clear(B) \wedge on(B,T)$
- The goal to have A on B, and B on C:
  $on(A,B) \wedge on(B,C)$

Planning Actions

- An action schema can be specified by:
  - *name* and *parameters* of action
  - *preconditions*: what positive literals must be true
  - *effects*: what becomes true and false
- Action schema to move block x from y to z:

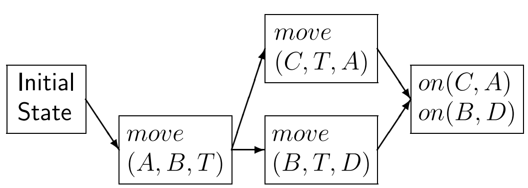
  $Action(move(x, y, z),$
  $\text{PRECOND}: clear(x) \wedge on(x, y) \wedge clear(z)$
  $\text{EFFECT}: clear(y) \wedge on(x, z) \wedge on(x, y) \wedge clear(z))$
- Create a complete, correct set of action schema for the blocks world.

Partial Order Planning 5

Introduction to Partial Order Planning

- Progression and regression planning require that actions be totally ordered.
- Partial order planning only specifies those orderings that are needed.
- Example: Blocks-world with a table T and four blocks, A, B, C, and D.
- $Init(clear(A) \wedge on(A, B) \wedge on(B, T) \wedge clear(C)$
  $\wedge\, on(C, T) \wedge clear(D) \wedge on(D, T))$
  $Goal(on(C, A) \wedge on(B, D))$

Example



Partial Order Causal-Link Planning

A partial-order plan consists of the following:

- A set of steps. Start step, operators, finish step. The start and finish steps encode the initial state and goals.
- A set of orderings between pairs of steps.
- A set of causal links. Each causal link goes from an effect of one step to a precondition of another step.

Fixing Flaws

A flaw in a partial-order plan is:

- a precondition that is not supported by a causal link, or
- a causal link that is threatened by another step (the threat).

A flaw can be fixed by:

- adding a causal link, possibly adding an operator to support the causal link, or
- ordering the threat before the causal link ("demotion") of after ("promotion").

Planning Graphs 9

Introduction to Planning Graphs

- A *planning graph* is a sequence of levels corresponding to "time steps," alternating between "state levels" S_i and "action levels" A_i.
- It starts with state level S_0, which contains the literals true of the initial state.
- An action is in A_i if its preconds. are in S_i. It has edges from its preconds. in S_i and to its effects in S_{i+1}.
- Also, each literal in S_i has a persistence edge to the same literal in S_{i+1}.

Mutual Exclusion (Mutex) Links

- Two actions in A_i have a mutex link if a precondition or effect of one action conflicts with a precondition or effect of the other action.
- Two literals in S_{i+1} have a mutex link if one negates the other or if every pair of actions in A_i achieving them are mutex.
- An action cannot be in A_i if any two preconds. in S_i are mutex.

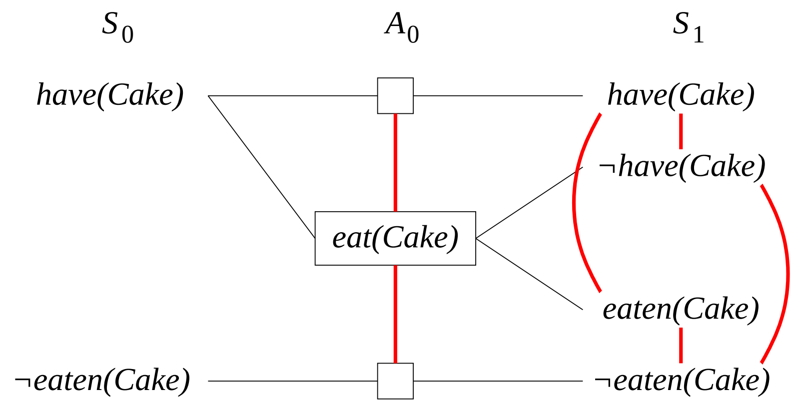
"Simple" Planning Graph Example

$Init(have(Cake))$
$Goal(have(Cake) \wedge eaten(Cake))$

$Action(eat(Cake),$
 PRECOND: $have(Cake)$
 EFFECT: $eaten(Cake) \wedge \neg have(Cake))$

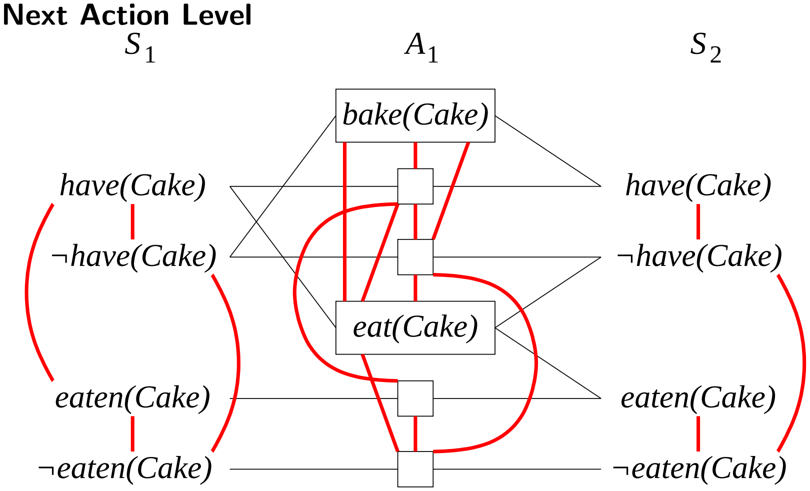
$Action(bake(Cake),$
 PRECOND: $\neg have(Cake)$
 EFFECT: $have(Cake))$

Initial Action Level



$have(Cake)$ and $eaten(Cake)$ in S_1 are mutex because keeping $have(Cake)$ is mutex with $eat(Cake)$.

Next Action Level



Both actions to achieve $\neg have(Cake)$ are mutex with the one action to achieve $\neg eaten(Cake)$.

Graphplan Algorithm

The main idea is to extract a plan from a planning graph. A search problem is defined by:

1. A state is a subset of literals in a state level.
2. There is an edge from a subset of S_i to a subset of S_{i-1} if there is a mutex-free subset of A_{i-1} with respective effects and preconds.
3. The initial state is the set of goals on the last level S_n of the planning graph.
4. The goal is to reach S_0.