Planning Graphs

1 Introduction to Planning Graphs

Planning graphs are used to represent a planning problem, aiding in heuristic estimation and directly helping in constructing plans. Planning graphs only work for propositional problems.

2 Structure of Planning Graphs

A planning graph is a sequence of levels, where:

- **Level 0** is the initial state.
- Each level contains a set of literals and a set of actions, representing what might be possible at that level.

At each level:

- **Literals** represent propositions that might be true.
- **Actions** represent possible actions at that level.

3 Example Problem: Have Cake and Eat It Too

3.1 Initial and Goal States

- **Initial State:** Have(Cake)
- **Goal State:** Have(Cake) $\wedge$ Eaten(Cake)

3.2 Actions

- **Action: Eat(Cake)**
  - **Precondition:** Have(Cake)
  - **Effect:** $\neg$Have(Cake) $\wedge$ Eaten(Cake)
- **Action: Bake(Cake)**
  - **Precondition:** $\neg$Have(Cake)
  - **Effect:** Have(Cake)

4 Graph Construction Process

4.1 Level 0 (S_0)

This is the initial state, which includes:

- Have(Cake)
- ¬Eaten(Cake)

4.2 Action Layer (A_0)

From S_0, we generate all applicable actions based on the preconditions being met:

- Eat(Cake)

4.3 Level 1 (S_1)

Apply the effects of actions in A_0 to generate the next state, including persistence actions (no-op) that allow literals to persist:

- Have(Cake)
- Eaten(Cake)
- ¬Have(Cake)
- ¬Eaten(Cake)

4.4 Handling Mutual Exclusions (Mutex)

Mutual exclusion (mutex) represents conflicts between actions or literals, which arise when:

- One action negates the effect of another (inconsistency).
- The effects of one action negate the preconditions of the other (interference).
- The preconditions of one action are mutually exclusive with the preconditions of the other.

5 Example: Have Cake and Eat It Too (Graph Representation)

The graph grows as we apply actions and their effects across levels:

- S_0: Have(Cake), ¬Eaten(Cake)

- A_0: Apply actions Eat(Cake) and Bake(Cake)
- S_1: Resultant literals including Have(Cake), Eaten(Cake), and their negations
- A_1: Apply the corresponding actions for further progression

The process continues until the graph reaches a point where two consecutive levels are identical or have the same number of literals.