Graphplan/ SATPlan

Chapter 11.4-11.7

Some material adapted from slides by
Jean-Claude Latombe / Lise Getoor

GraphPlan

GraphPlan: Basic idea

- Construct a graph that encodes constraints on possible plans
- Use this “planning graph” to constrain search for a valid plan
- Planning graph can be built for each problem in a relatively short time

Planning graph

- Directed, leveled graph with alternating layers of nodes
- Odd layers ("**state levels**") represent candidate propositions that could possibly hold at step *i*
- Even layers ("**action levels**") represent candidate actions that could possibly be executed at step *i*, including maintenance actions [do nothing]
- **Arcs** represent preconditions, adds and deletes
- We can only execute one real action at any step, but the data structure keeps track of **all actions and states that are *possible***

GraphPlan properties

- STRIPS operators: conjunctive preconditions, no conditional or universal effects, no negations
  - Planning problem must be convertible to propositional representation
  - NO continuous variables, temporal constraints, …
  - Problem size grows exponentially
- Finds “shortest” plans (by some definition)
- Sound, complete, and will terminate with failure if there is no plan

What actions and what literals?

- Add an action in level A_i if ***all*** of its preconditions are present in level S_i
- Add a literal in level S_i if it is the effect of ***some*** action in level A_{i-1} (*including no-ops*)
- Level S_0 has all of the literals from the initial state

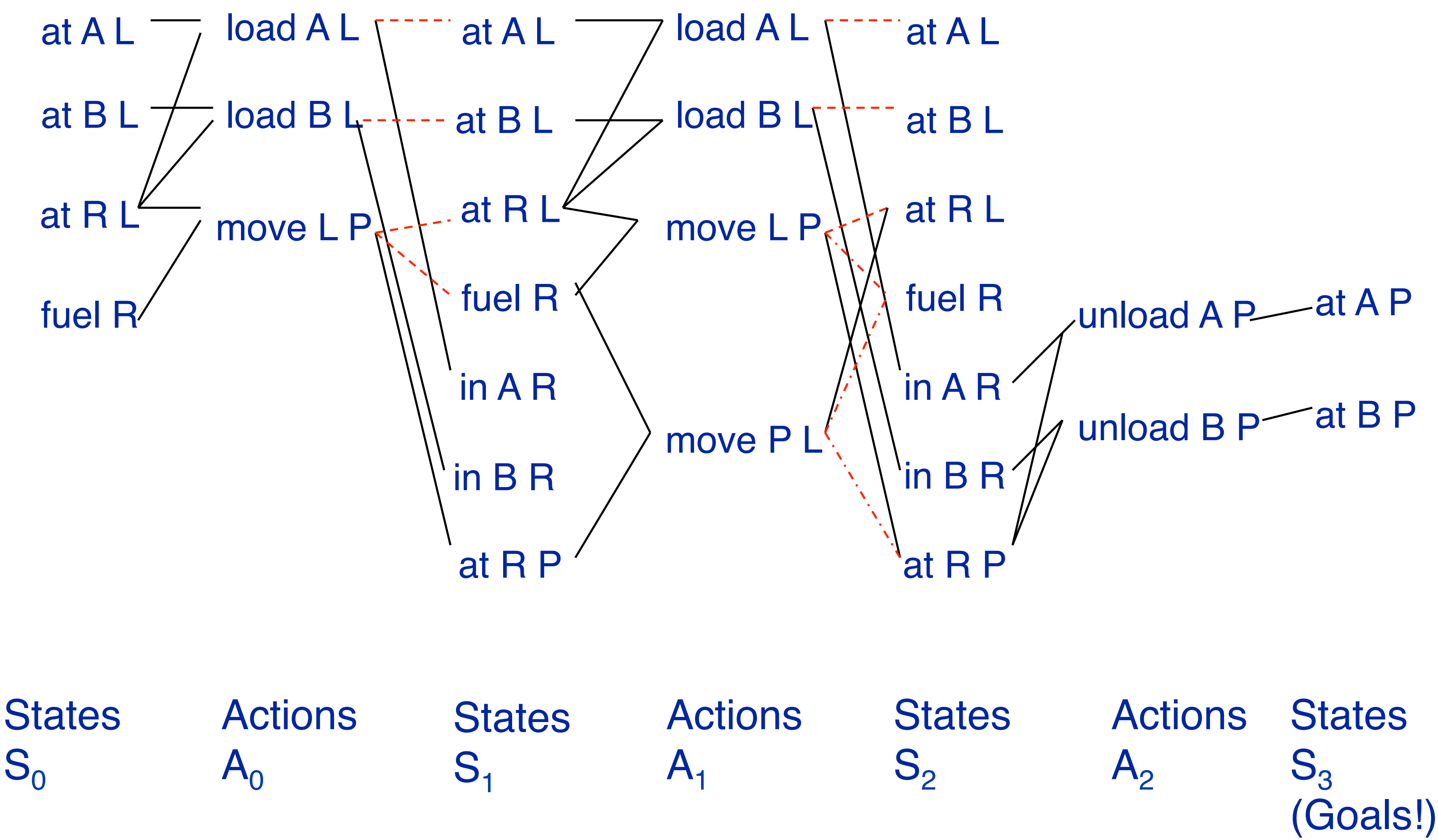
Simple domain: Ricket to Mars

- Literals:
  - at X Y — X is at location Y
  - fuel R — rocket R has fuel
  - in X R — X is in rocket R
- Actions:
  - load X L — load X (onto R) at location L (X and R must be at L)
  - unload X L — unload X (from R) at location L (R must be at L)
  - move X Y — move rocket R from X to Y (R must be at L and have fuel)
- Graph representation:
  - Solid black lines: preconditions/effects
  - Dotted red lines: negated preconditions/effects

```
(define (domain rockets)
  (:requirements :strips)
  (:predicates (cargo ?x) (rocket ?x) (location ?x)
            (at ?t ?l) (in ?c ?r) (fuel ?r))
  (:action load
   :parameters (?c ?r ?l)
   :precondition (and (cargo ?c) (rocket ?r) (location ?l)
                   (at ?c ?l) (at ?r ?l))
   :effect (and (not (at ?c ?l)) (in ?c ?r)))
  (:action unload
   :parameters (?c ?r ?l)
   :precondition (and (cargo ?c) (rocket ?r) (location ?l)
                   (in ?c ?r) (at ?r ?l))
   :effect (and (not (in ?c ?r)) (at ?c ?l)))
  (:action fly
   :parameters (?r ?dep ?dst)
   :precondition (and (rocket ?r) (location ?dep) (location ?dst)
                   (at ?r ?dep) (fuel ?r))
   :effect (and (not (at ?r ?dep)) (at ?r ?dst) (not (fuel ?r)))))
```

```
(define (problem rrt5)
  (:domain rockets)
  (:requirements :strips)
  (:objects venus earth mars moon saturn x1 x2 x3 x4 x5
        anna beth carol diane emma fiona)
  (:init
   (location venus) (location earth) (location mars) (location moon)
   (location saturn) (rocket x1) (rocket x2) (rocket x3) (rocket x4)
   (rocket x5) (cargo anna) (cargo beth) (cargo carol) (cargo diane)
   (cargo emma) (cargo fiona)
   (at x1 venus) (at x2 earth) (at x3 mars) (at x4 moon) (at x5 saturn)
   (at anna venus) (at beth venus) (at carol earth) (at diane mars)
   (at emma moon) (at fiona saturn)
   (fuel x1) (fuel x2) (fuel x3) (fuel x4) (fuel x5))
  (:goal (and (at anna earth) (at beth saturn) (at carol mars)
         (at diane moon) (at emma saturn) (at fiona earth))))
```

Example planning graph



BlackBox Planner

STRIPS-based plan representation

↓

Planning graph

↓

CNF representation

↓

CSP/SAT solver

↓

CSP solution

↓

Plan

Whew!