Classical Planning

Shiwali Mohan

Computer Science and Engineering
University of Michigan

May 25, 2012

Outline

Review

- What is a plannning problem?

Review

- What is a plannning problem?
  - asks if we can reach a goal state from the initial state
  - how?

Review

- What is a plannning problem?
  - asks if we can reach a goal state from the initial state
  - how?
- Environment is
  - fully observable
  - deterministic
  - finite
  - static
  - discrete

Review

- What is a plannning problem?
  - asks if we can reach a goal state from the initial state
  - how?
- Environment is
  - fully observable
  - deterministic
  - finite
  - static
  - discrete
- Combination of
  - Logic (state and action representation)
  - Search (search for generating a plan)

Review

- What is a plannning problem?
  - asks if we can reach a goal state from the initial state
  - how?
- Environment is
  - fully observable
  - deterministic
  - finite
  - static
  - discrete
- Combination of
  - Logic (state and action representation)
  - Search (search for generating a plan)
- State Representation

Review

- What is a plannning problem?
  - asks if we can reach a goal state from the initial state
  - how?
- Environment is
  - fully observable
  - deterministic
  - finite
  - static
  - discrete
- Combination of
  - Logic (state and action representation)
  - Search (search for generating a plan)
- State Representation
  - Conjunction of ground, functionless terms (fluents)
    - examples: Poor ˆ Unknown; At(Truck1, Melbourne)
  - Close world assumption

Review

- What is a plannning problem?
  - asks if we can reach a goal state from the initial state
  - how?
- Environment is
  - fully observable
  - deterministic
  - finite
  - static
  - discrete
- Combination of
  - Logic (state and action representation)
  - Search (search for generating a plan)
- State Representation
  - Conjunction of ground, functionless terms (fluents)
    - examples: Poor ˆ Unknown; At(Truck1, Melbourne)
  - Close world assumption
    - fluents that are not mentioned are false
    - objects with unique names are distinct

Review

- What is a plannning problem?
  - asks if we can reach a goal state from the initial state
  - how?
- Environment is
  - fully observable
  - deterministic
  - finite
  - static
  - discrete
- Combination of
  - Logic (state and action representation)
  - Search (search for generating a plan)
- State Representation
  - Conjunction of ground, functionless terms (fluents)
    - examples: Poor ˆ Unknown; At(Truck1, Melbourne)
  - Close world assumption
    - fluents that are not mentioned are false
    - objects with unique names are distinct
  - `at(Truck1,y)?`, `-Wealthy?`, `At(Father(Fred),Sydney)`, `On(blockA, blockB) V On (blockA, blockC)`

Review

- Action Representation
  - as action schemas
  - single schema represents a set of ground actions

```
Action(Fly(p, from, to),
   PRECOND: At(p,from) ^ Plane (p) ^ Airport (from)
            ^ Airport (to)
   EFFECT: -At(p,from) ^ At(p,to))
```

  - Universally quantified
  - applicable in states where all the preconditions are satisfied
  - delete list: remove fluents that appear as negative literals in effects
  - add list: add fluents that appear as positive literals in effects

PDDL Formulation

- What predicates would you use to represent the problem?

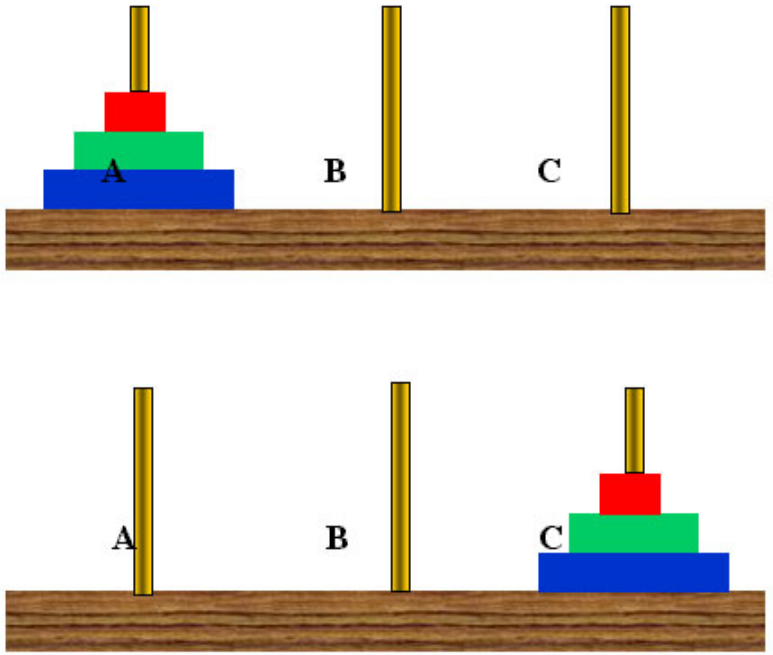


Figure: objects: disks - red, green, blue; pegs - A, B, C

PDDL Formulation

- What predicates would you use to represent the problem?
  - `clear()`, `on()`, `smaller()`, `disk()`

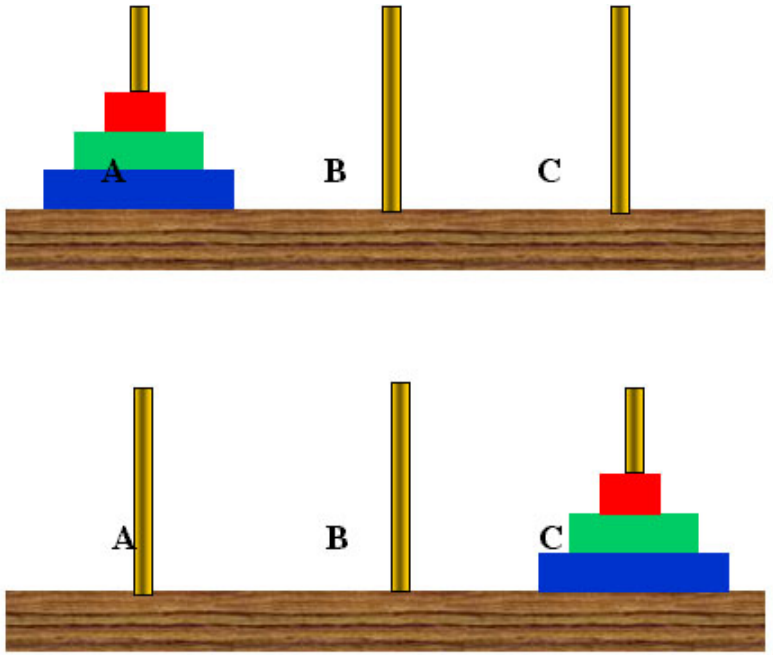


Figure: objects: disks - red, green, blue; pegs - A, B, C

PDDL Formulation

- What predicates would you use to represent the problem?
  - `clear()`, `on()`, `smaller()`, `disk()`
- What is the initial state?

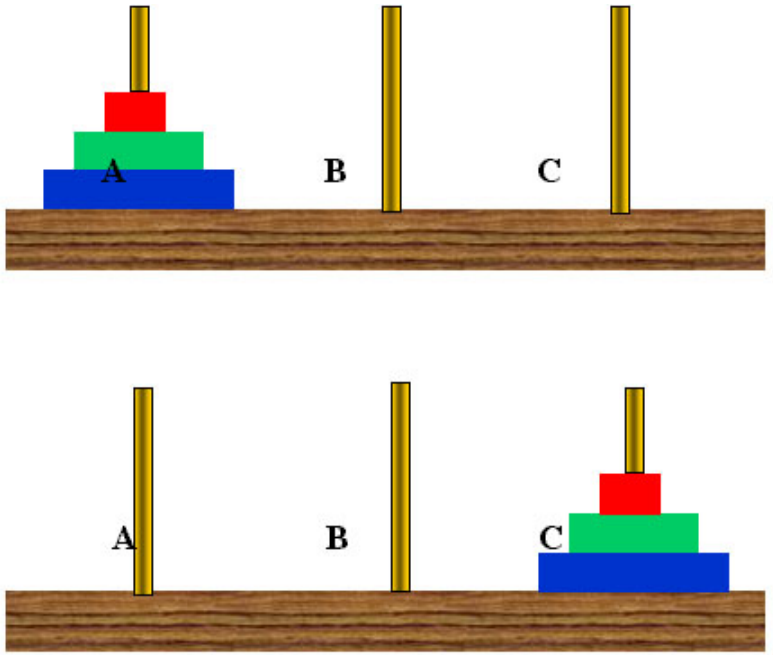


Figure: objects: disks - red, green, blue; pegs - A, B, C

PDDL Formulation

- What predicates would you use to represent the problem?
  - `clear()`, `on()`, `smaller()`, `disk()`
- What is the initial state?
  - `on(Red, Green)`, `on(Green, Blue)`, `on(Blue, A)`, `clear(Red)`, `disc(Red)` ... `clear(B)`, `clear(C)`, `smaller(Red, Green)` `smaller(Red, Blue)`, `smaller(Red, A)` ...

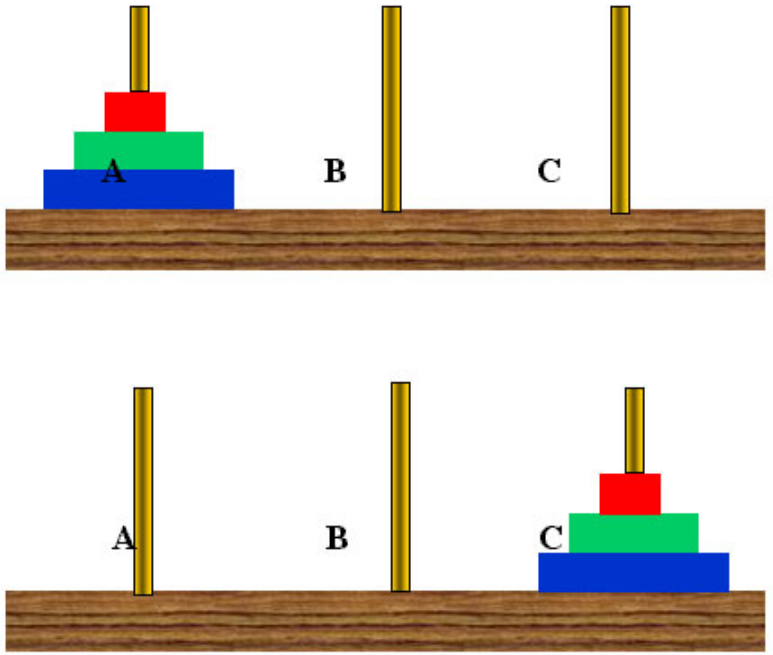


Figure: objects: disks - red, green, blue; pegs - A, B, C

PDDL Formulation

- What predicates would you use to represent the problem?
  - `clear()`, `on()`, `smaller()`, `disk()`
- What is the initial state?
  - `on(Red, Green)`, `on(Green, Blue)`, `on(Blue, A)`, `clear(Red)`, `disc(Red)` ... `clear(B)`, `clear(C)`, `smaller(Red, Green)` `smaller(Red, Blue)`, `smaller(Red, A)` ...
- What is the goal state?

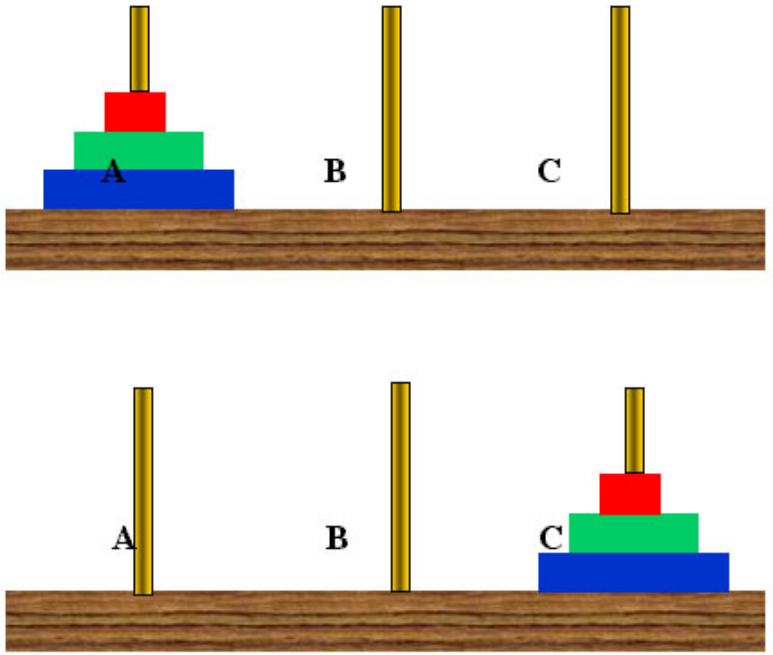


Figure: objects: disks - red, green, blue; pegs - A, B, C

PDDL Formulation

- What predicates would you use to represent the problem?
  - `clear()`, `on()`, `smaller()`, `disk()`
- What is the initial state?
  - `on(Red, Green)`, `on(Green, Blue)`, `on(Blue, A)`, `clear(Red)`, `disc(Red)` ... `clear(B)`, `clear(C)`, `smaller(Red, Green)` `smaller(Red, Blue)`, `smaller(Red, A)` ...
- What is the goal state?
  - `on(Red, Green)`, `on(Green, Blue)`, `on(Blue, C)`

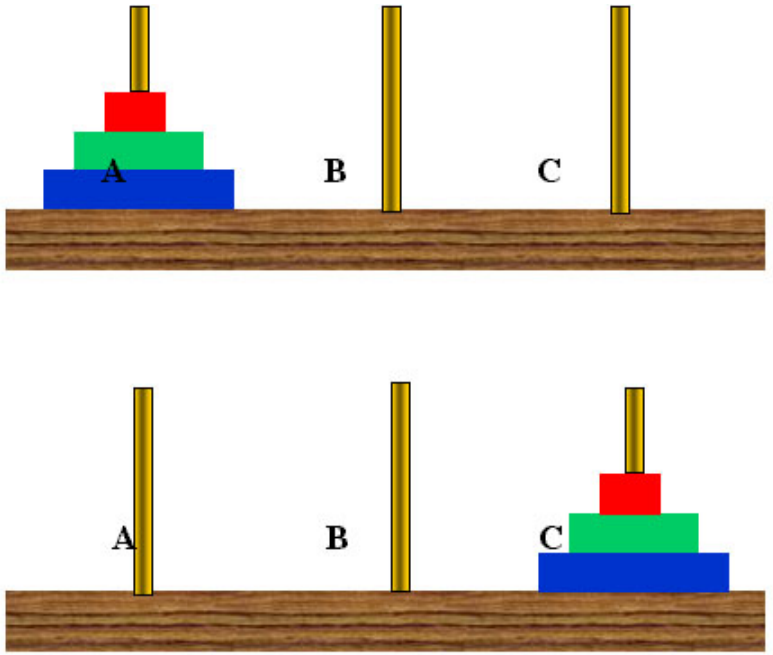


Figure: objects: disks - red, green, blue; pegs - A, B, C

PDDL Formulation

- What predicates would you use to represent the problem?
  - `clear()`, `on()`, `smaller()`, `disk()`
- What is the initial state?
  - `on(Red, Green)`, `on(Green, Blue)`, `on(Blue, A)`, `clear(Red)`, `disc(Red)` ... `clear(B)`, `clear(C)`, `smaller(Red, Green)` `smaller(Red, Blue)`, `smaller(Red, A)` ...
- What is the goal state?
  - `on(Red, Green)`, `on(Green, Blue)`, `on(Blue, C)`
  - How are actions defined?

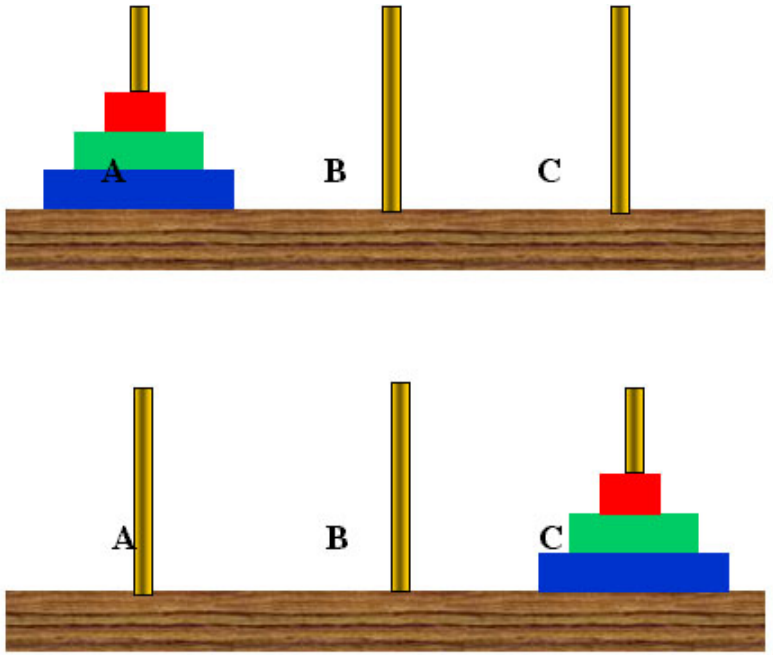


Figure: objects: disks - red, green, blue; pegs - A, B, C

PDDL Formulation

- What predicates would you use to represent the problem?
  - `clear()`, `on()`, `smaller()`, `disk()`
- What is the initial state?
  - `on(Red, Green)`, `on(Green, Blue)`, `on(Blue, A)`, `clear(Red)`, `disc(Red)` ... `clear(B)`, `clear(C)`, `smaller(Red, Green)` `smaller(Red, Blue)`, `smaller(Red, A)` ...
- What is the goal state?
  - `on(Red, Green)`, `on(Green, Blue)`, `on(Blue, C)`

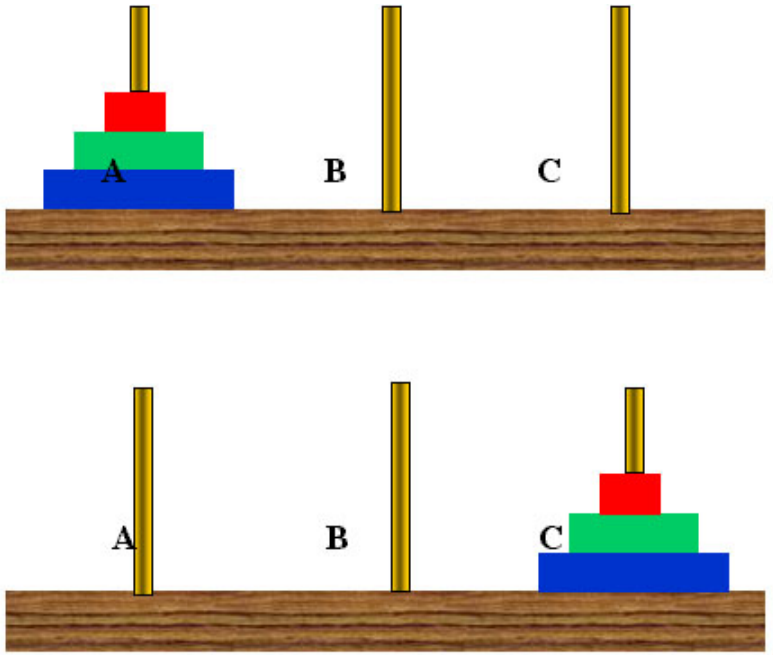
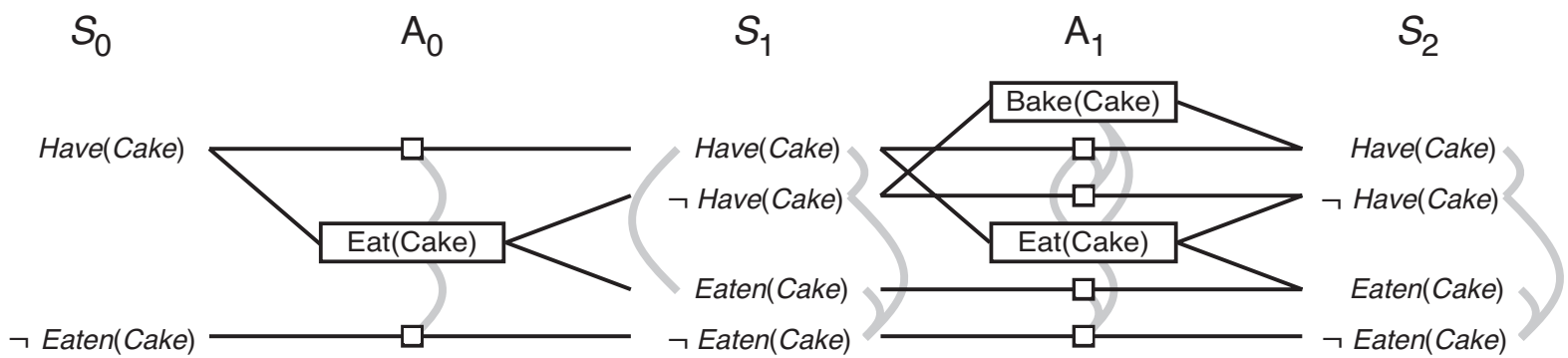


Figure: objects: disks - red, green, blue; pegs - A, B, C

- How are actions defined?
  - `Action(move(disk, source, destination)`
    `PRECOND: clear(disk) ^ on(disk, source) ^ clear(destination)`
    `^smaller(disk,destination)`
    `EFFECT: on(disk, destination) ^ -on(disk, source) ^`
    `-clear(destination) ^clear(source)`

Planning Graph Review

- Directed graph
  - organized into levels
  - level S_0 is the initial state that consists of fluents that hold in S_0
  - level A_0 consisting of actions applicable in S_0.
  - Alternate S_i and A_i until termination
  - persistence actions and mutex

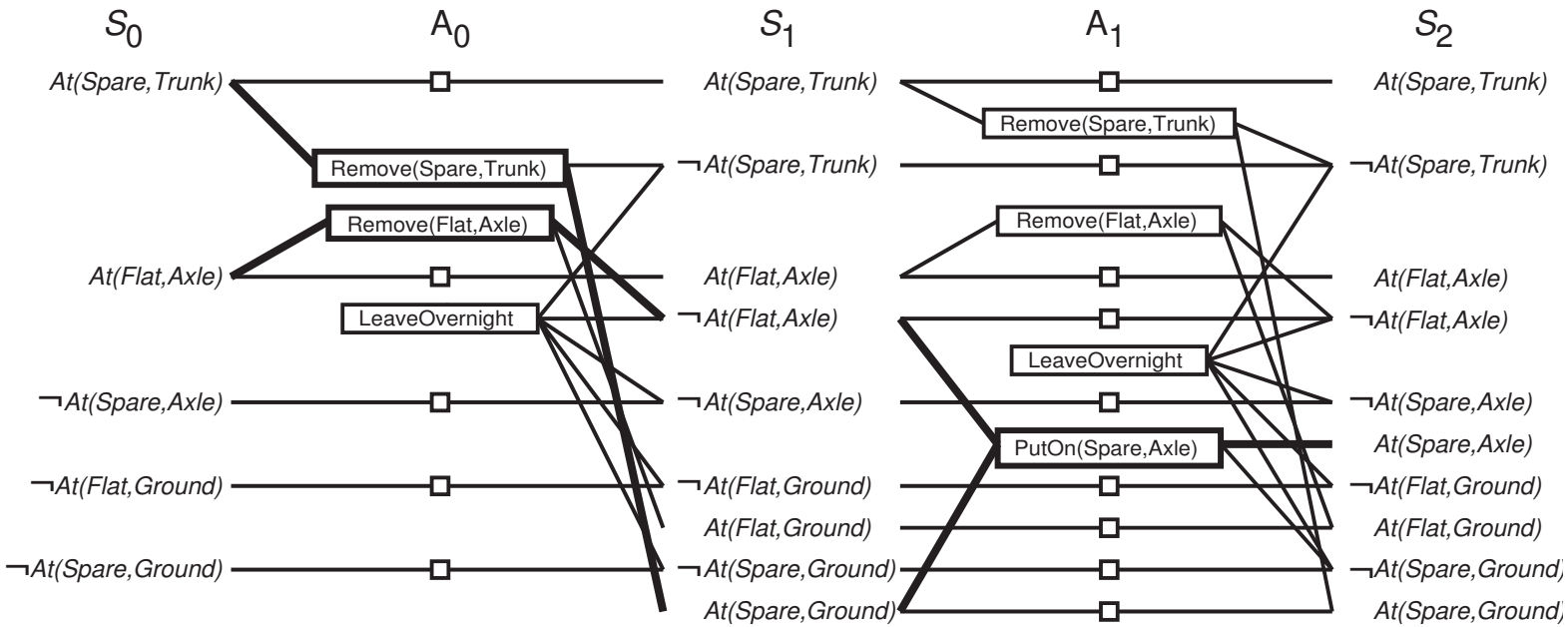


- Uses
  - for generating plans
  - for heuristic estimation

Spare Tire Example (in book)

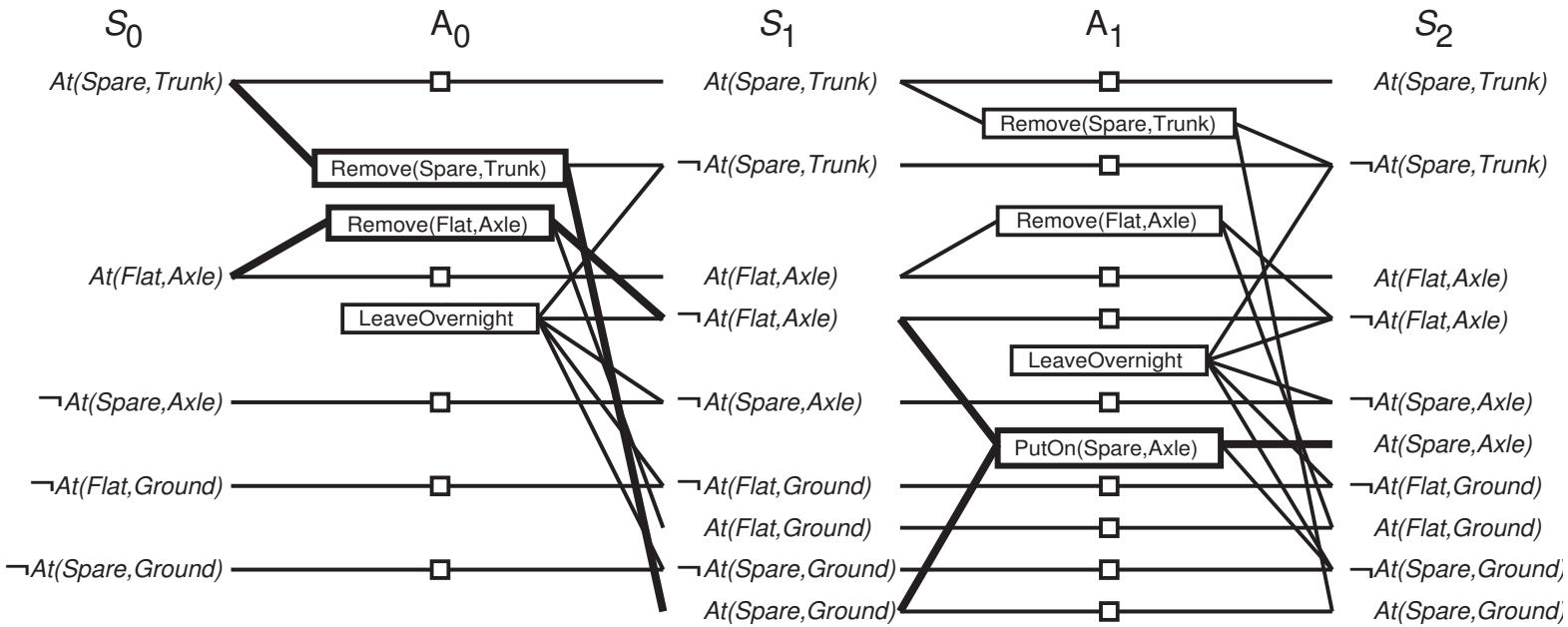
```
Init(Tire(Flat) ^ Tire(Spare) ^ At(Flat,Axle) ^ at(Spare,Trunk))
Goal(At(Spare,Axle))
Action(Remove(obj,loc),
    PRECOND: At(obj,loc)
    EFFECT: -At(obj,loc) ^ At(obj,ground))
Action(PutOn(t,Axle),
    PRECOND: Tire(t) ^ At(t,Ground) ^ -At(Flat,Axle)
    EFFECT: -At(t,Ground) ^ At(t,Axle)
Action(LeaveOvernight)
    PRECOND:
    EFFECT: -At(Spare,Ground) ^ -At(Spare,Axle) ^ -At(Spare,Trunk)
            ^-At(Flat,Ground) ^ -At(Flat,Axle) ^-At(Flat,Trunk)
```

Spare Tire Example (in book)



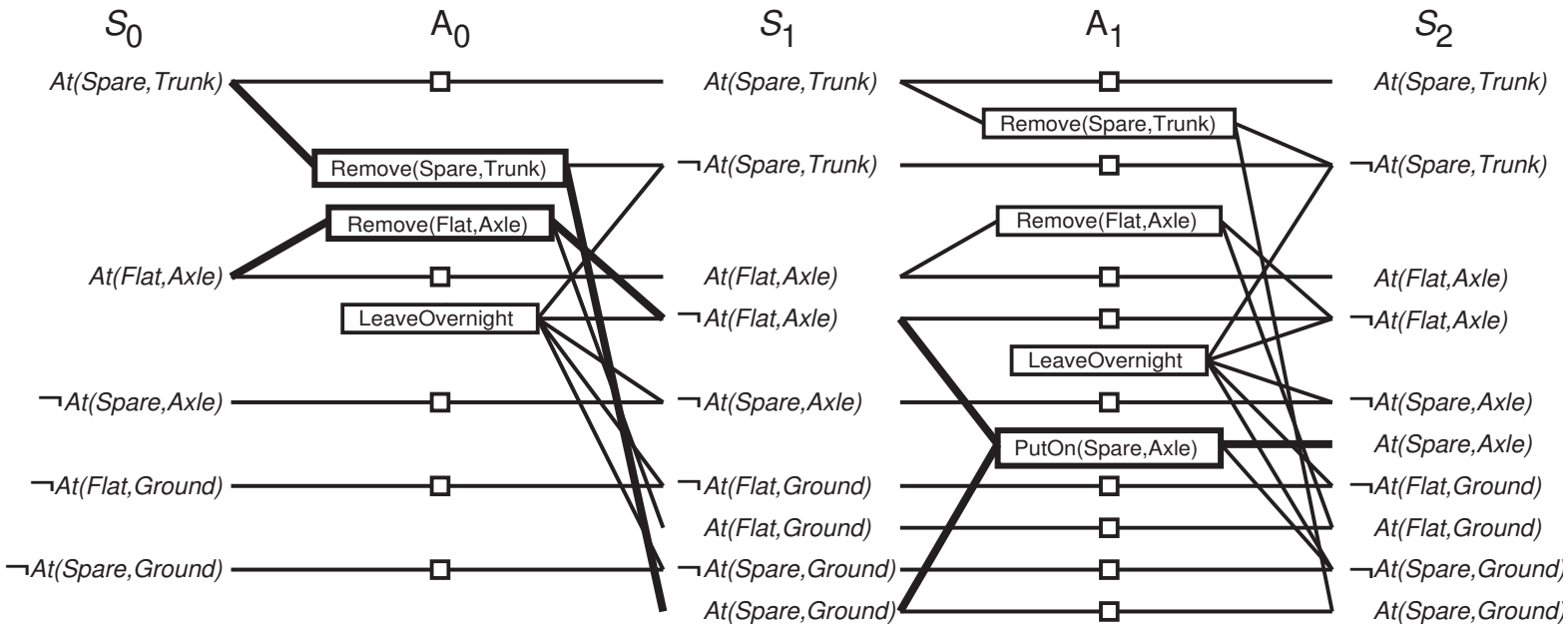
- Mutex actions?

Spare Tire Example (in book)



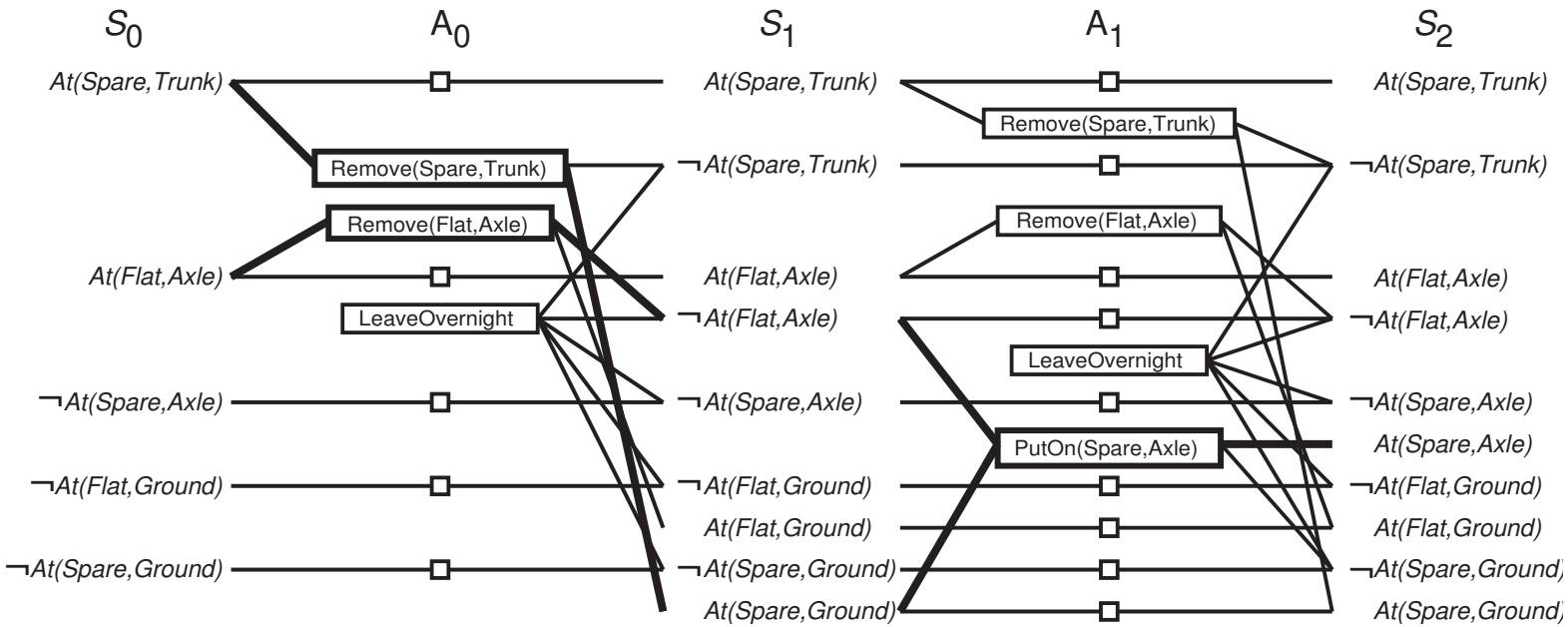
- Mutex actions?
  - Inconsistent effects?

Spare Tire Example (in book)



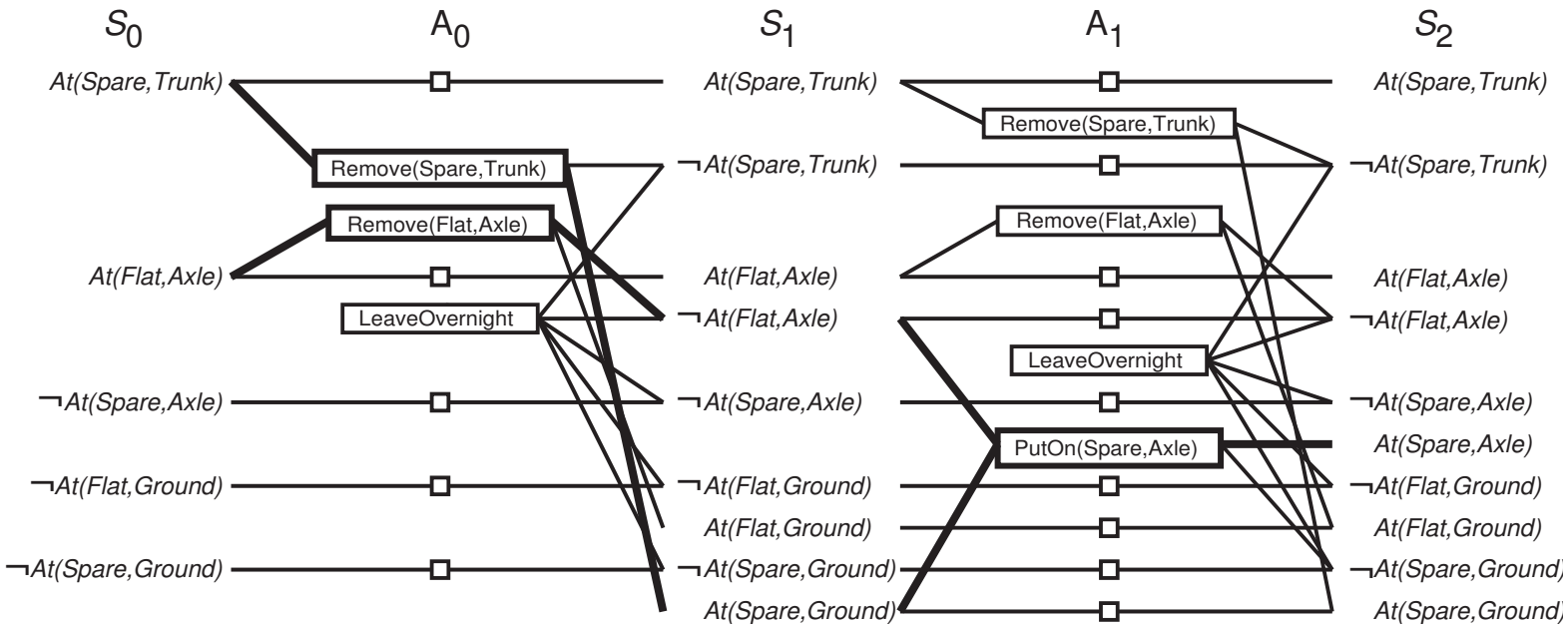
- Mutex actions?
  - Inconsistent effects?
    - one action negates an effect of the other

Spare Tire Example (in book)



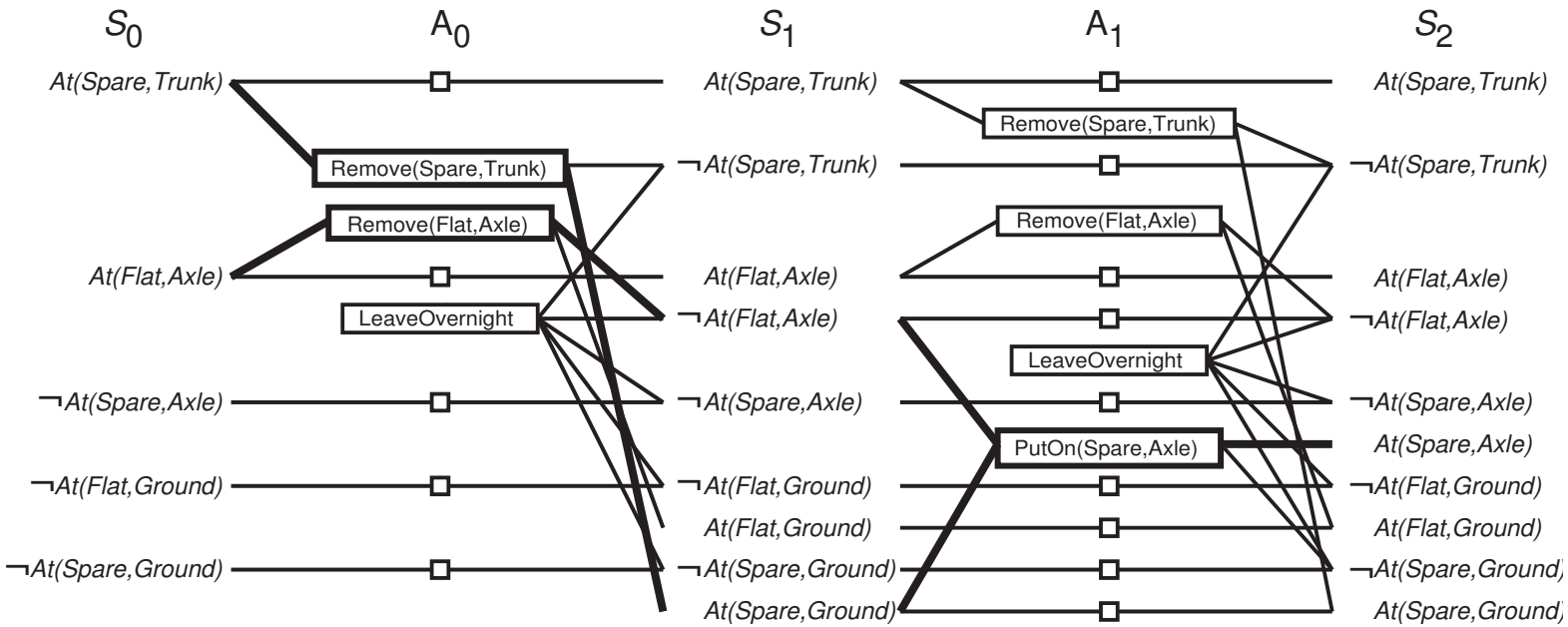
- Mutex actions?
  - Inconsistent effects?
    - one action negates an effect of the other
  - Interference?

Spare Tire Example (in book)



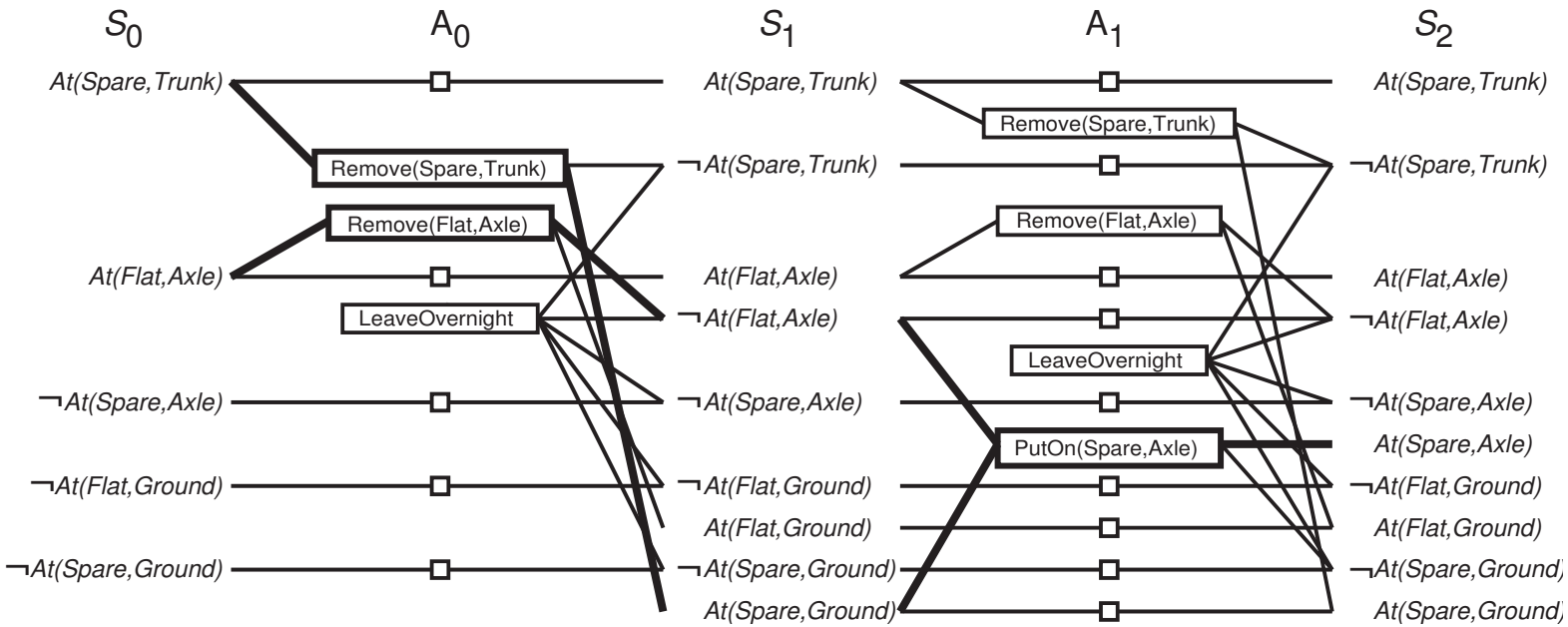
- Mutex actions?
  - Inconsistent effects?
    - one action negates an effect of the other
  - Interference?
    - one of the effects of an action is the negation of a precondition of the other

Spare Tire Example (in book)



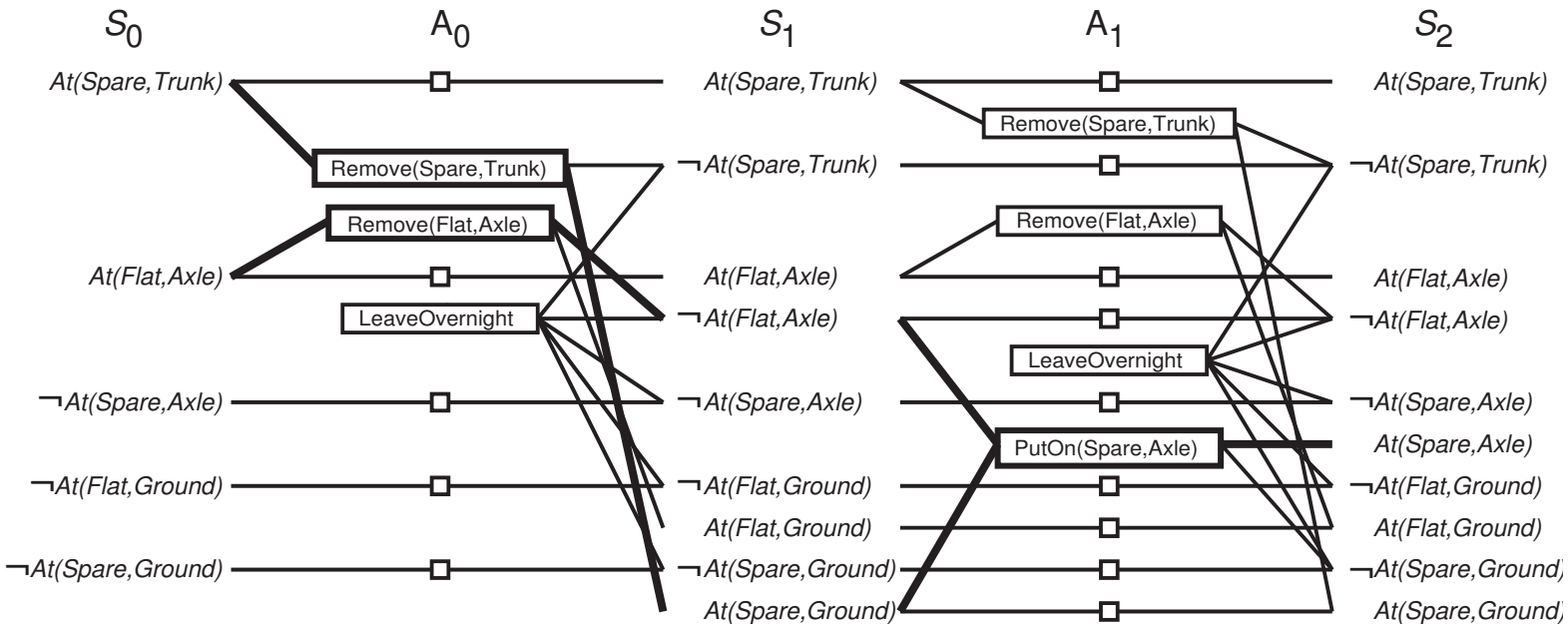
- Mutex actions?
  - Inconsistent effects?
    - one action negates an effect of the other
  - Interference?
    - one of the effects of an action is the negation of a precondition of the other
  - Competing needs?

Spare Tire Example (in book)



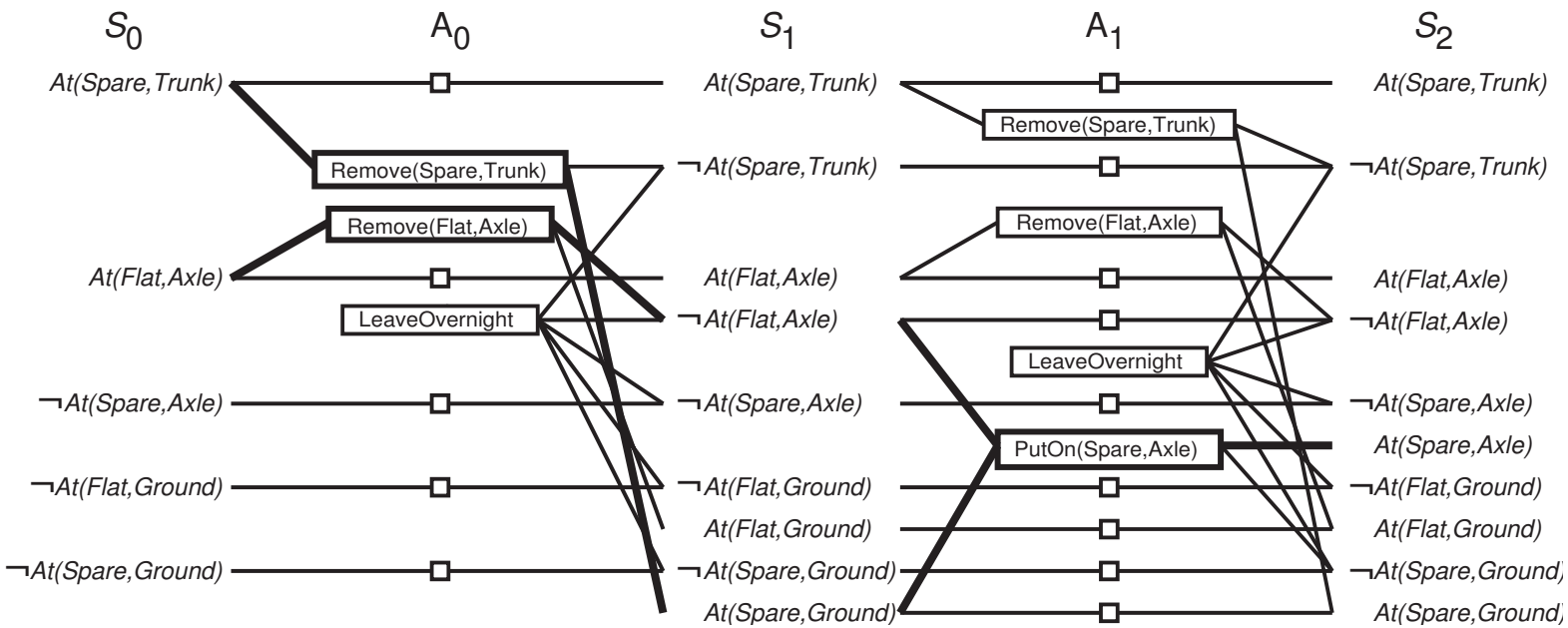
- Mutex actions?
  - Inconsistent effects?
    - one action negates an effect of the other
  - Interference?
    - one of the effects of an action is the negation of a precondition of the other
  - Competing needs?
    - One of the preconditions of one action is mutually exclusive with a precondition of the other

Spare Tire Example (in book)



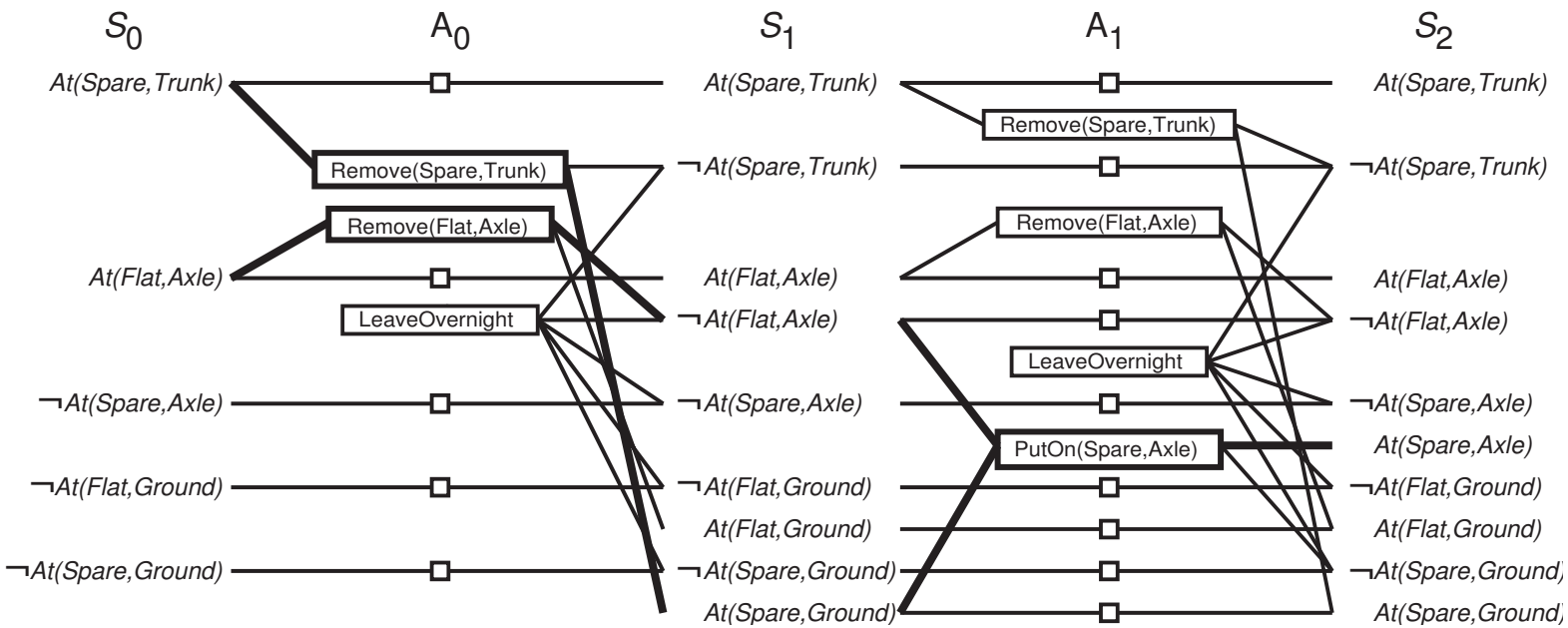
- Mutex actions?
  - Inconsistent effects?
    - one action negates an effect of the other
    - `Remove(Spare,Trunk)` with `LeaveOvernight`
  - Interference?
    - one of the effects of an action is the negation of a precondition of the other
    - `Remove(Flat,Axle)` with `LeaveOvernight`
  - Competing needs?
    - One of the preconditions of one action is mutually exclusive with a precondition of the other
    - `PutOn(Spare,Axle)` with `Remove(Flat,Axle)`

Spare Tire Example (in book)



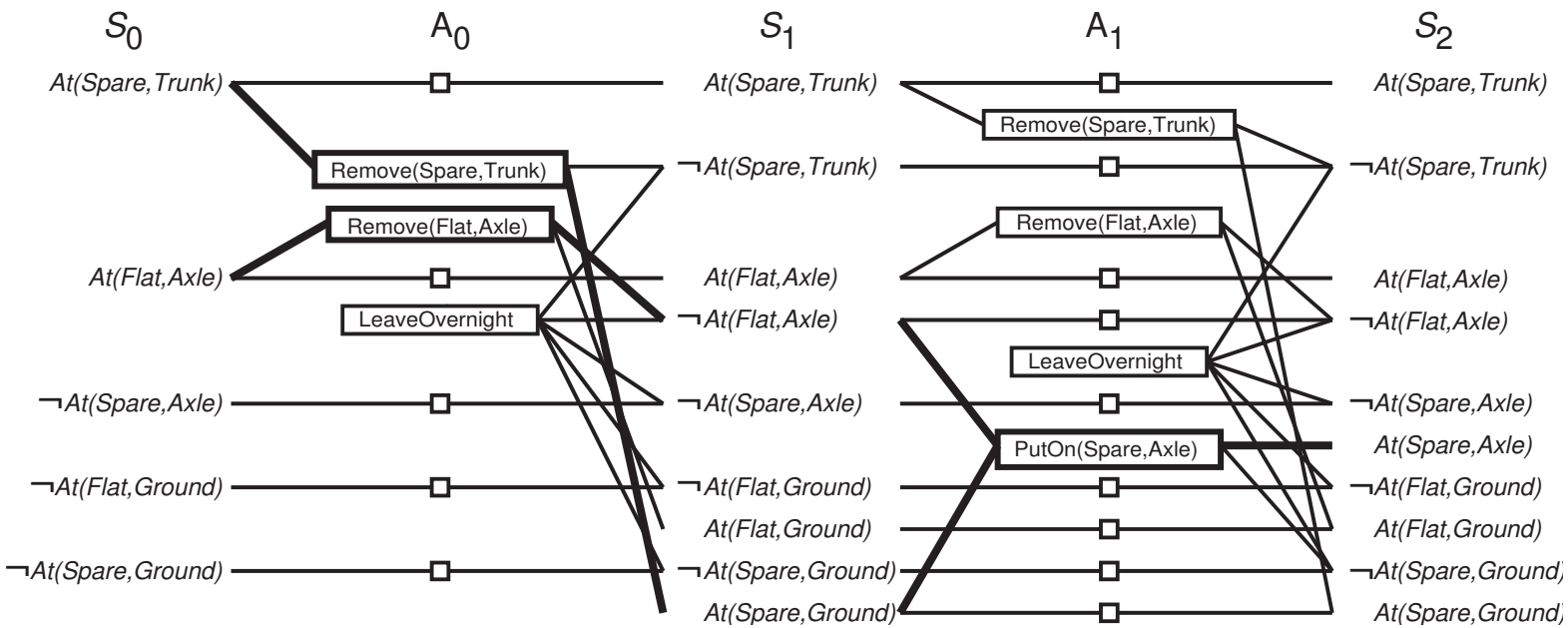
- Mutex literals?
  - Negated literal?

Spare Tire Example (in book)



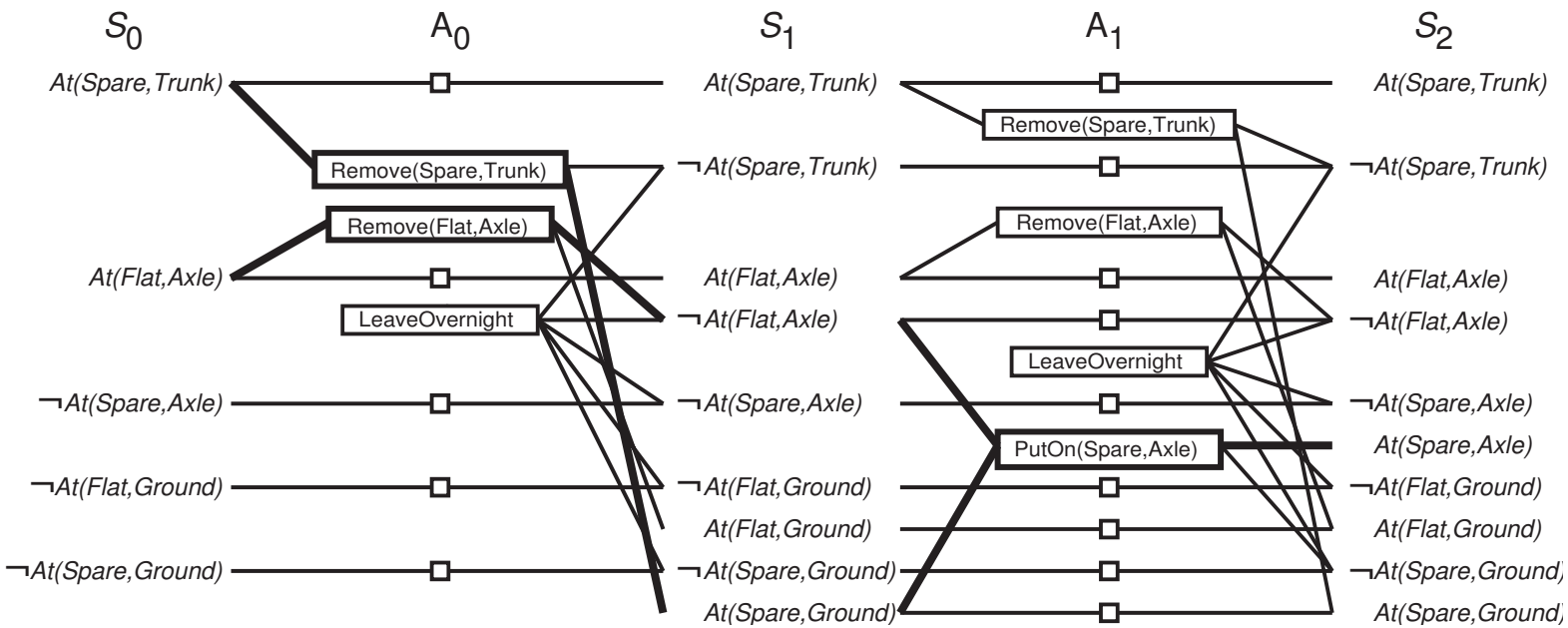
- Mutex literals?
  - Negated literal?
  - Inconsistent support?

Spare Tire Example (in book)



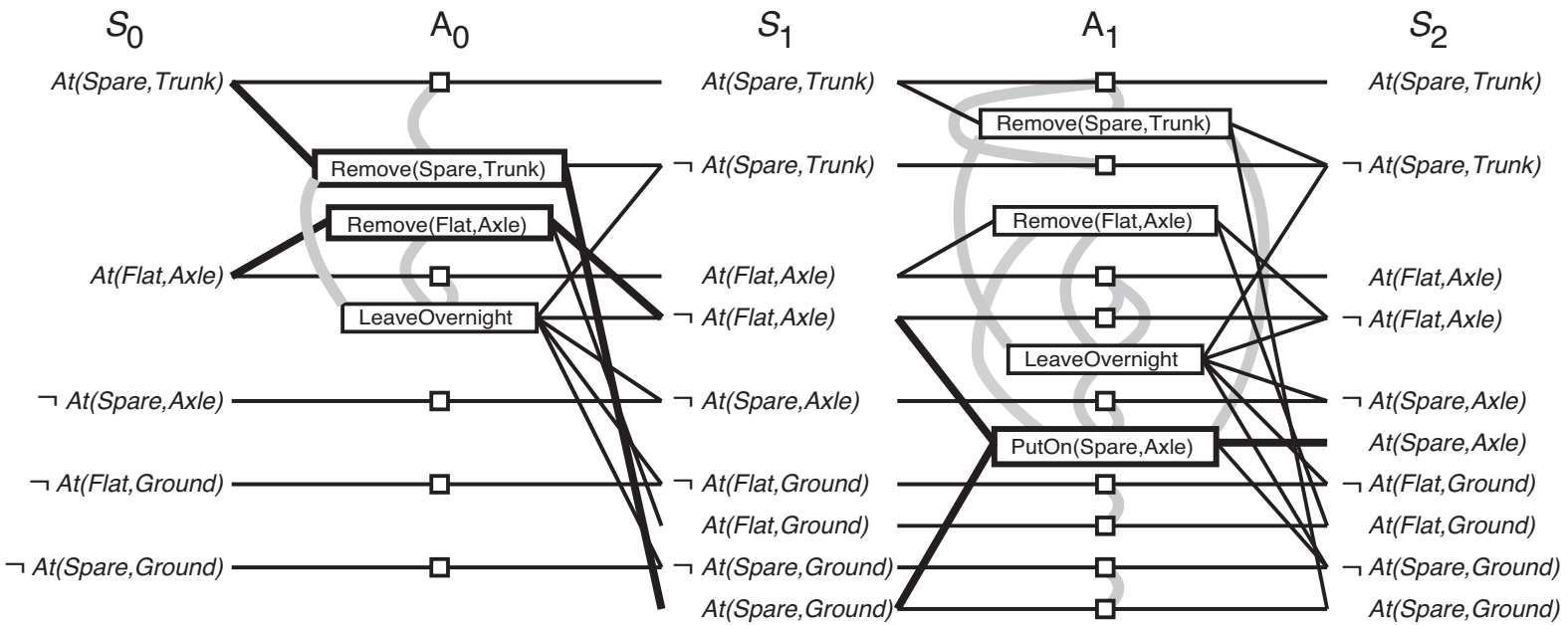
- Mutex literals?
  - Negated literal?
  - Inconsistent support?
    - if each possible pair of actions that achieve the two literal is mutex

Spare Tire Example (in book)



- Mutex literals?
  - Negated literal?
  - Inconsistent support?
    - if each possible pair of actions that achieve the two literal is mutex
    - `At(Spare, Axle)` with `At(Flat,Axle)` in S2

Extract Solution



- Extract Solution can be formulated as CSP or backward search
- As a CSP
  - variables? values?
  - constraints?
- As backward search
  - Initial state?
  - Goal?
  - Actions?
  - Cost