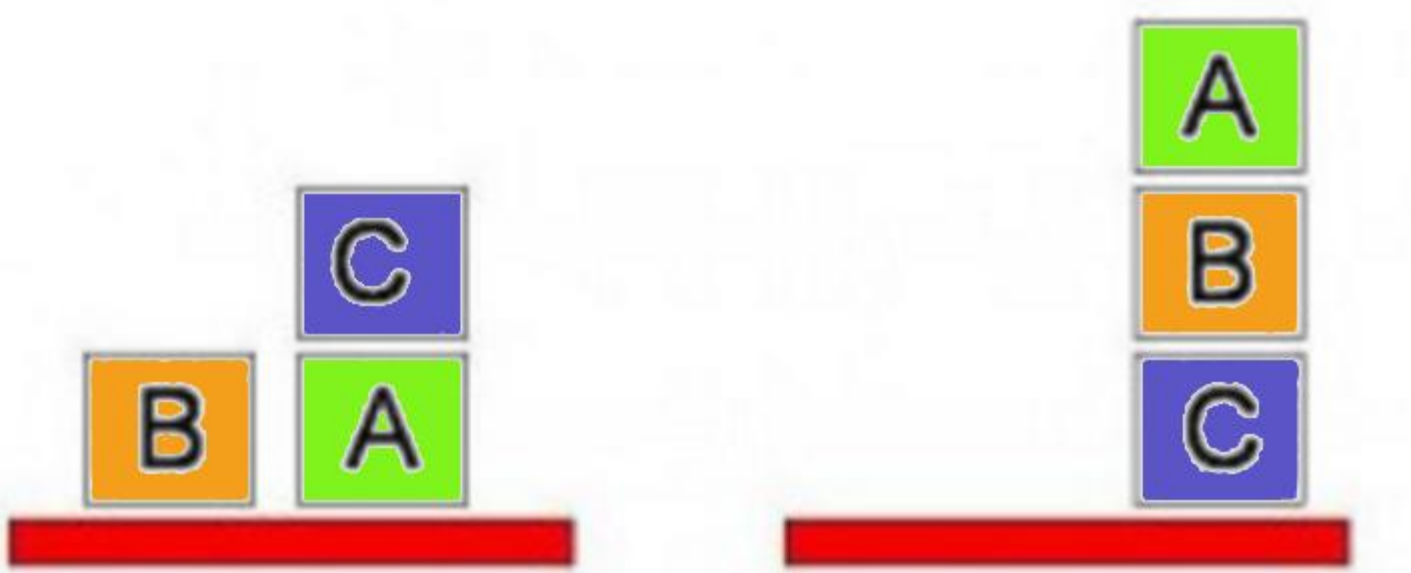
Solving Planning Problems



Homework 6

For question 6, the universal quantifier applies to the whole sentence:

- $\forall x\ \text{Stench}(x) \Rightarrow \exists\ y\ \text{Adjacent}(y,x) \wedge \text{Wumpus}(y)$

The above is equivalent to:

- $\forall x\ (\text{Stench}(x) \Rightarrow (\exists\ y\ \text{Adjacent}(y,x) \wedge \text{Wumpus}(y)))$

Homework 6

In question 7, the universal quantifiers apply to the whole sentence: :

- $\forall x \, \forall y \; Wumpus(x) \wedge Adjacent(x, y) \Rightarrow Stench(y)$

The above is equivalent to:

- $\forall x \, \forall y \; ((Wumpus(x) \wedge Adjacent(x, y)) \Rightarrow Stench(y))$

Homework 6

Resolution Proof:

We do not eliminate any sentences from the KB. We only generate new sentences.

All the sentences remain valid till the end.

We prove that KB KB $\wedge \neg\alpha$ is unsatisfiable by generating a sentence that is unsatisfiable.

A sentence is unsatisfiable when it is of the form: $A \wedge \neg A$

Today

How do we solve a planning problem represented in PDDL?

- State Space Search
  - Progression Search
  - Regression Search
  - Heuristics

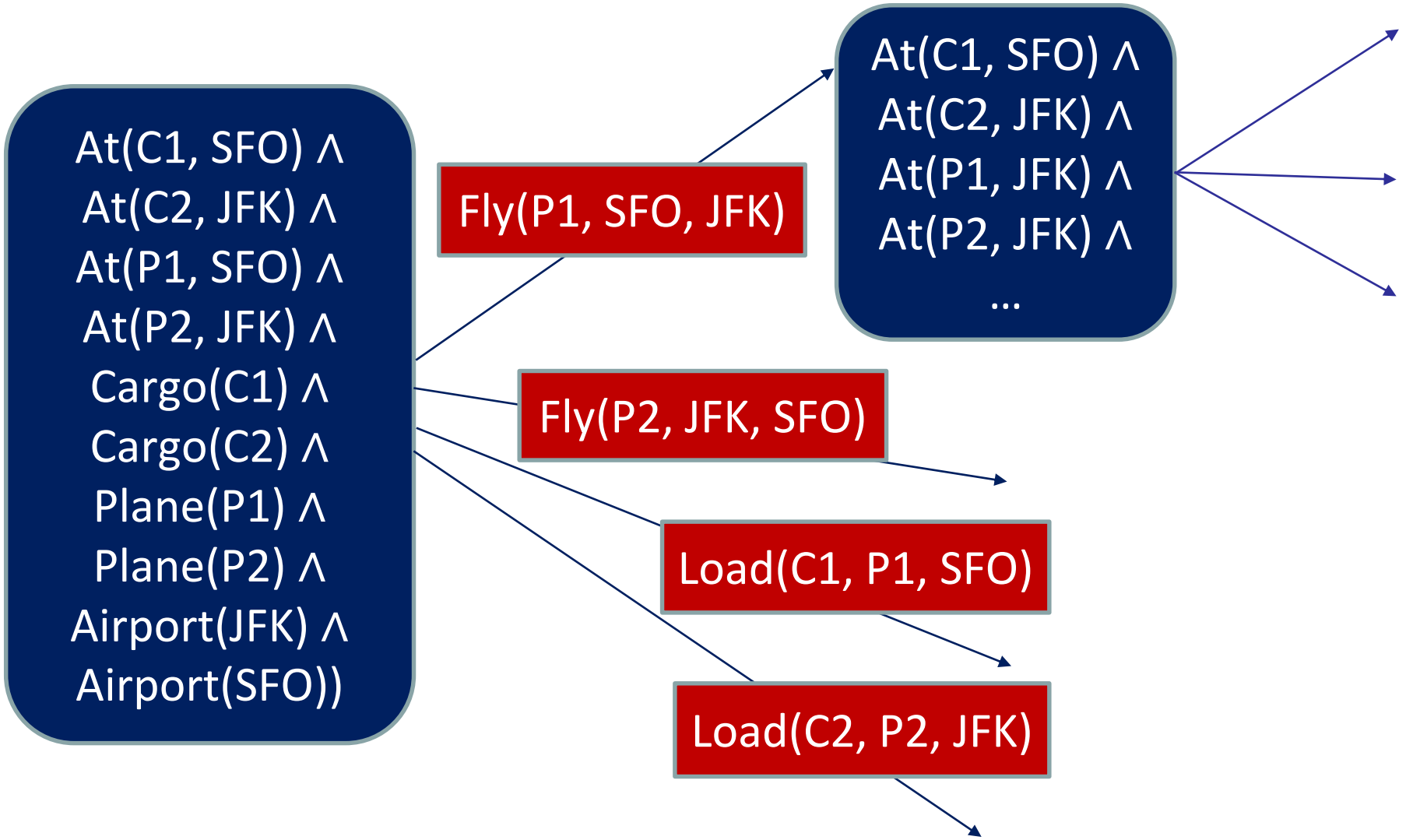
- Planning Graphs

Progression (Forward) State Space Search

Start with the initial state and apply each possible action repeatedly hoping to find a goal state.

- Unify state with preconditions to find applicable actions.
- Use effects to construct next state: (s−Del(a))UAdd(a)

Progression (Forward) State Space Search

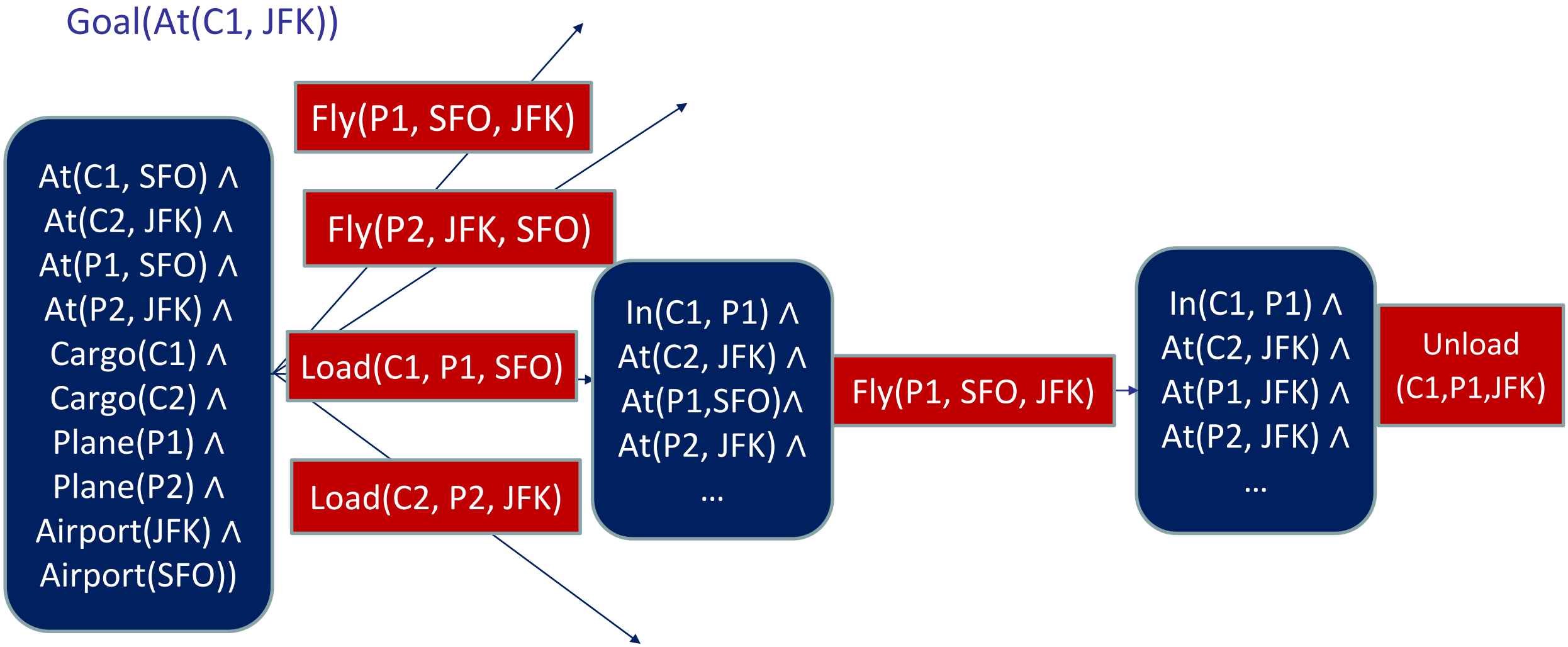


Progression State Space Search

Problem with progression search?

- exploring irrelevant actions
- in this example we should not fly any planes before loading the cargo.

Progression State Space Search



Progression Search – Irrelevant Actions

Goal(Own(0136042597))

Init(ISBN(0136042597), ISBN(0136042598), ISBN(0136042599),
ISBN(0136042600),…)

Action(Buy(i),

PRECOND: ISBN(i)

EFFECT: Own(i))

Our goal is to buy the AIMA textbook.

How many applicable actions?

A. 0
B. 1
C. 2
D. 3
E. Billions

Progression Search – Irrelevant Actions

Goal(Own(0136042597))

Init(ISBN(0136042597), ISBN(0136042598), ISBN(0136042599),
ISBN(0136042600),…)

Action(Buy(i),

PRECOND: ISBN(i)

EFFECT: Own(i))

Our goal is to buy the AIMA textbook.

ISBNs are 10 or 13 digits, so this action schema has at least 10 billion ground actions (that's a huge branching factor).

An uninformed forward search will enumerate them all.

Progression State Space Search Feasibility?

Only with strong heuristics.

PDDL representation makes it easy to automatically generate domain-independent heuristics.

Regression Search

Start from the goal, and search backward until we reach a state implied by the initial state. Note that the goal does not represent a single state, but rather a family of states.

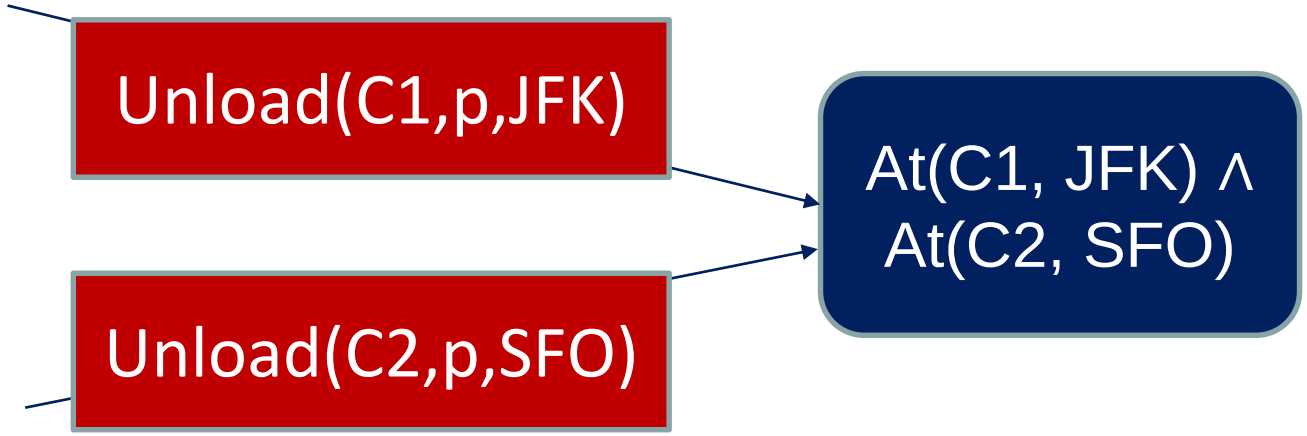
- Unify elements of state with effects to find relevant actions.
- Use preconditions to construct previous state:
  (s−Add(a))UPrecond(a)

Regression Search

At(C1, JFK) ∧
At(C2, SFO)

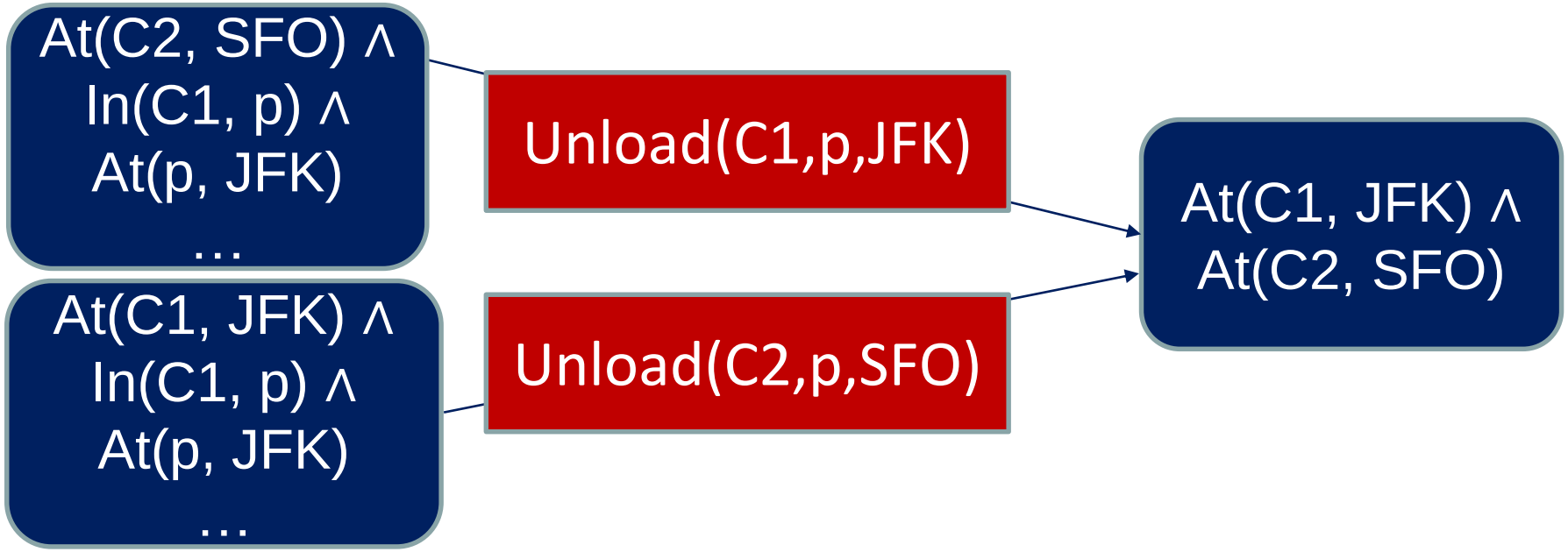
For an action to be relevant, one of its effects must unify with an element of the goal.

Regression Search



Action(Unload(c, p, a),
PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
EFFECT: At(c, a) ∧ ¬ In(c, p))

Regression Search

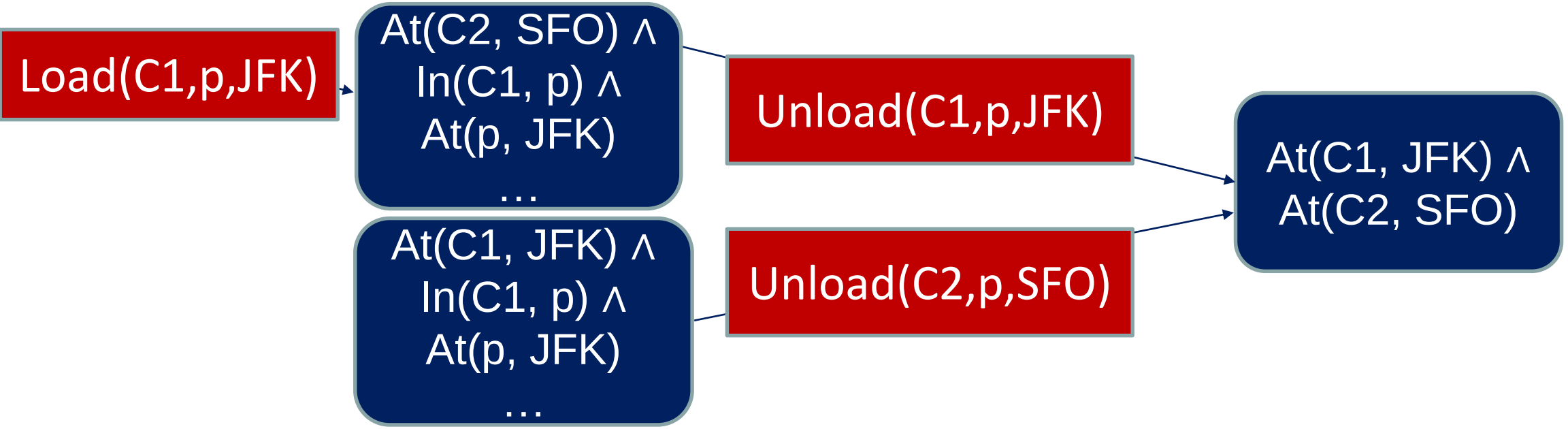


Action(Unload(c, p, a),

PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)

EFFECT: At(c, a) ∧ ¬ In(c, p))

Regression Search



Action(Load(c, p, a),

PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)

EFFECT: ¬ At(c, a) ∧ In(c, p))

Regression Search

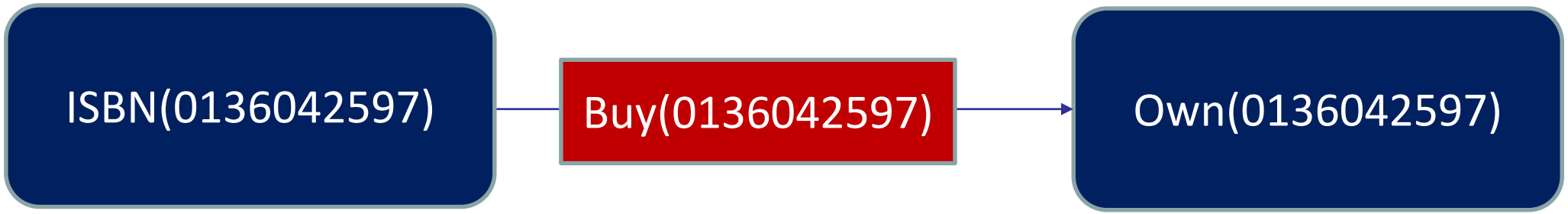
Goal(Own(0136042597))

Init(ISBN(0136042597), ISBN(0136042598), ISBN(0136042599),
ISBN(0136042600),…)

Action(Buy(i),

PRECOND: ISBN(i)

EFFECT: Own(i))



For an action to be relevant, one of its effects must unify with an element of the goal.

Heuristics for Planning

PDDL representation makes it easy to automatically generate heuristics.

Heuristics are exact solutions to a relaxed problem.

To relax the problem:

- Increase number of possible actions – make actions easier
  - Ignore preconditions
  - Ignore negative effects (delete lists)
- Reduce number of states
  - Abstract states (merge multiple states into one)
- Decomposition

Heuristics – Ignore Preconditions

Remove preconditions from action schemas:

- All actions become applicable
- Every element in the goal is achievable in one step
- Number of steps for the solution is the number of unsatisfied goal elements?

Problem?

- Some actions may achieve several elements (literals)
- Some action may undo the effect of another action

Solution?

- In addition to ignoring preconditions, ignore effects that are not in the goal.

We can also remove selected preconditions from action schemas

Heuristics – Ignore Negative Effects

Remove all negative effects of actions

- no action may undo the effects of another
- monotonic progression toward the goal
- known as the ignore delete list heuristic

Heuristics – State Abstraction

Map several ground states onto one abstraction:

- Ignore some literals

Example: if our goal involves C1 and C2 only: ignore At fluents not involving C1 and C2.

Heuristics – Decomposition

Subgoal independence Assumption:

- cost of solving a conjunction of subgoals is approximated by the sum of the costs of solving each subgoal independently
- Optimistic
  - Where subplans interact negatively (the block problem)
- Pessimistic (not admissible)
  - Redundant actions in subplans can be replaced by a single action in merged plan. We can use the maximum of the costs instead of the sum

Planning Graphs

The main disadvantage of state-space search is the size of the search tree (exponential).

The planning graph is a polynomial size approximation of the complete tree.

A planning graph is used to:

1. Derive heuristics
2. Extract a solution to the planning problem: GRAPHPLAN algorithm

Planning Graphs

A planning graph is a sequence $\langle S_0, A_0, S_1, A_1, \ldots, S_i \rangle$ of levels

- Alternating state levels & action levels
- Levels correspond to time stamps
- Start at the initial state
- State level is a set of literals (all the literals that could be true at that level)
- Action level is a set of ground actions (all those actions whose preconditions appear in the preceding state level)

Planning Graphs

Init(Have(Cake))

Goal(Have(Cake)∧Eaten(Cake))

Action(Eat(Cake)

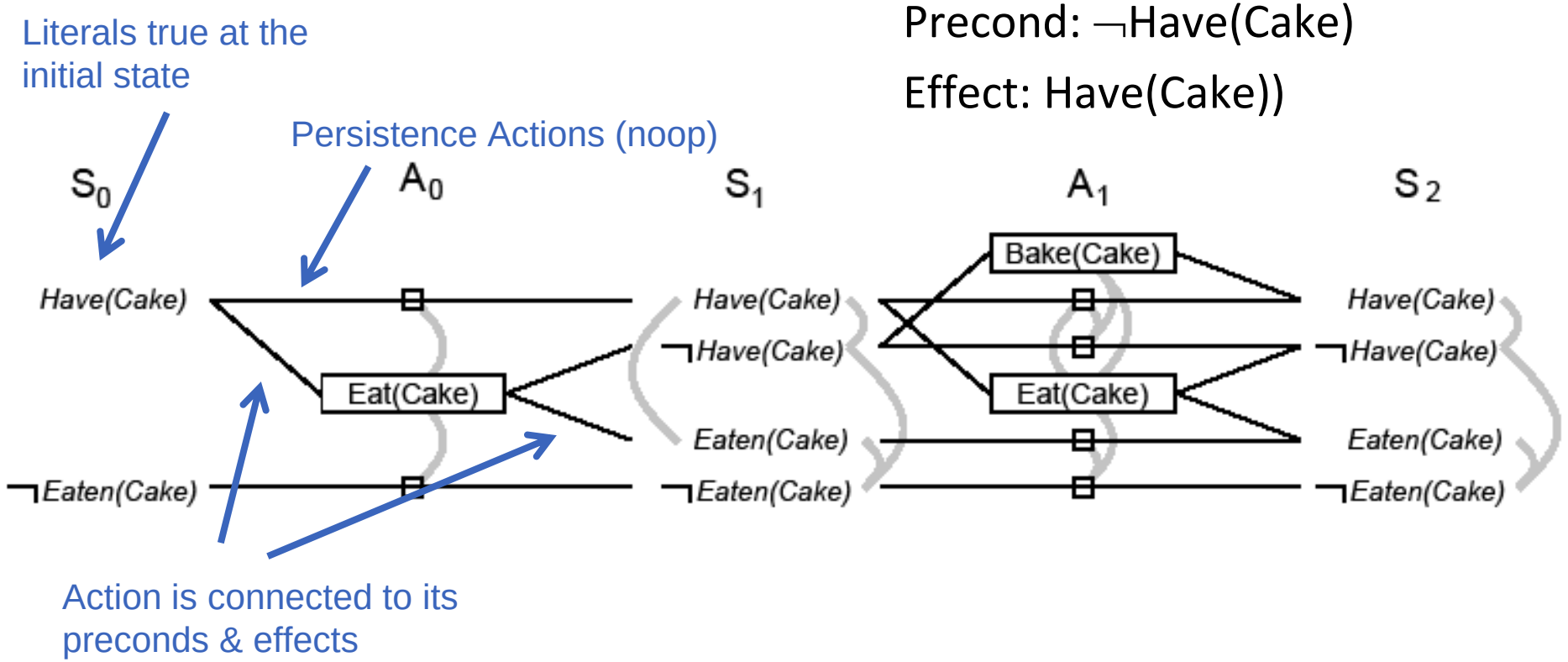
Precond: Have(Cake)

Effect: ¬Have(Cake)∧Eaten(Cake))

Action(Bake(Cake)

Precond: ¬Have(Cake)

Effect: Have(Cake))



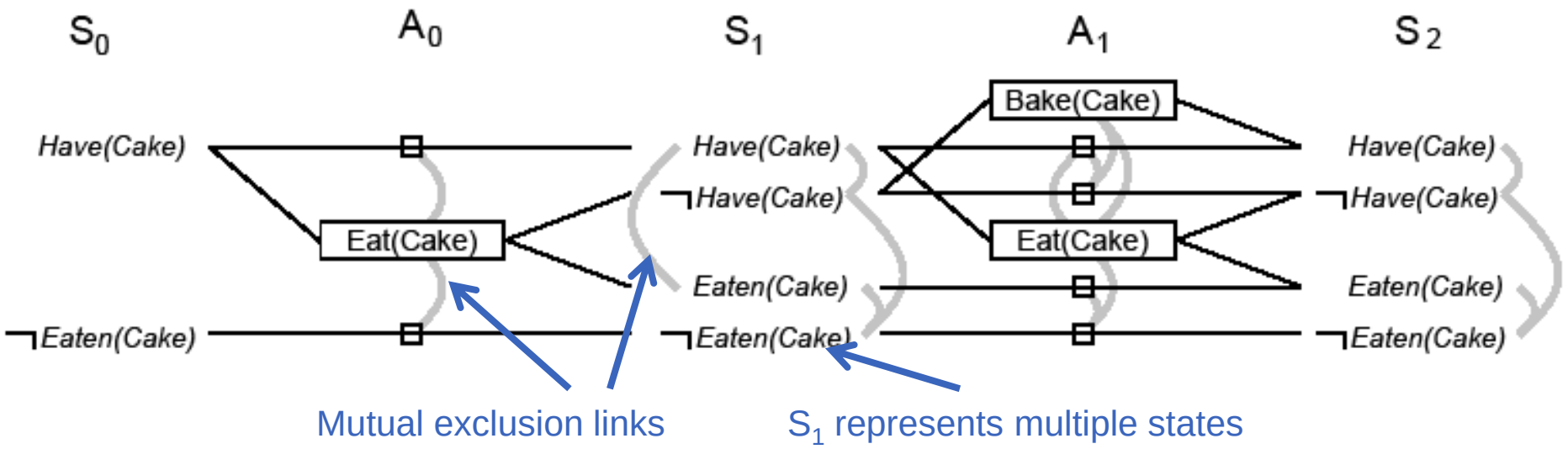
Planning Graph

At each state level, list all literals that may hold at that level

At each action level, list applicable actions and noops

Repeat until plan 'levels off,' no new literals appears (S_i=S_{i+1})

Add mutual exclusion (mutex) links between levels: conflicting actions and conflicting literals



Mutex Links between Actions

Inconsistent effects: one action negates an effect of another

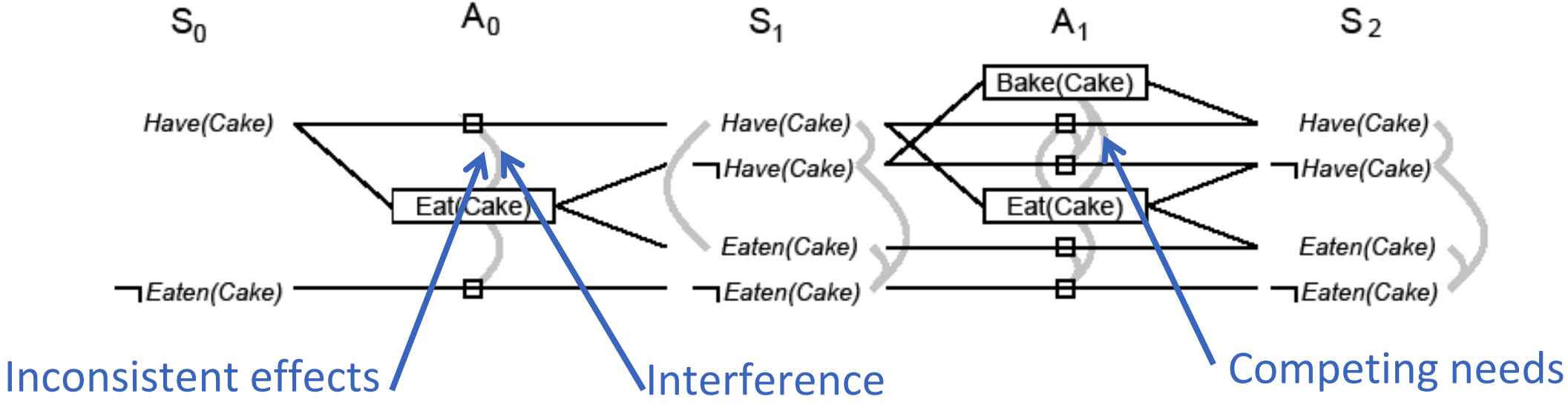
- Eat(Cake) & noop of Have(Cake) disagree on effect Have(Cake)

Interference: An action effect negates the precondition of another

- Eat(Cake) negates precondition of the noop of Have(Cake):

Competing needs: A precondition on an action conflicts with precondition of another

- Bake(Cake) & Eat(Cake): compete on Have(Cake) precondition



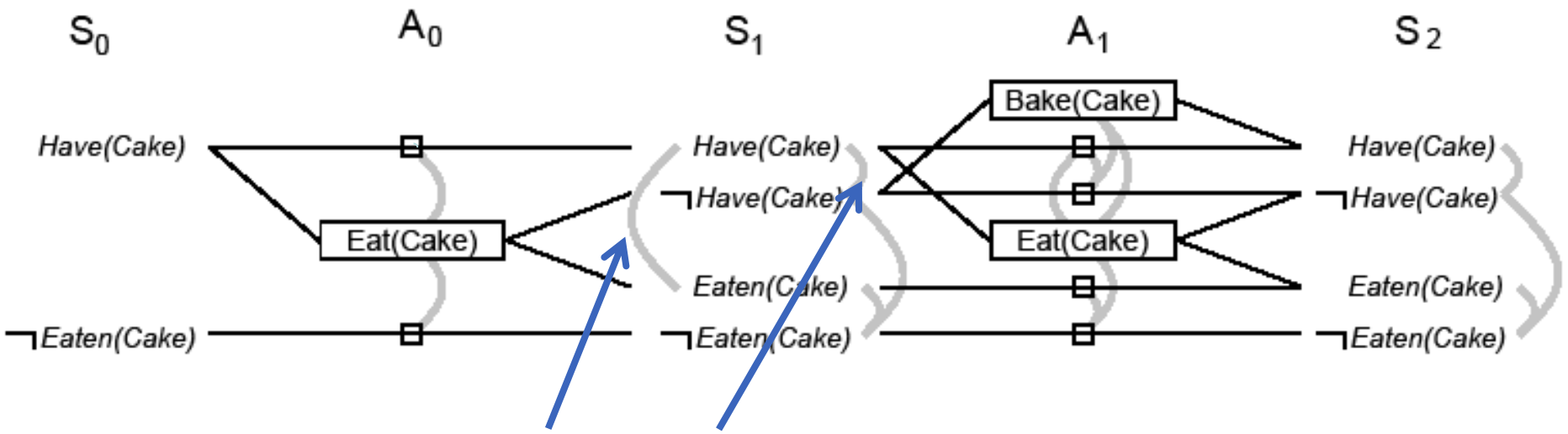
Mutex Links between Literals

Two literals are negation of each other

Inconsistent support: Each pair of actions that can achieve the two literals is mutex

In S1, Have(Cake) & Eaten(Cake) are mutex

In S2, they are not because Bake(Cake) & the noop of Eaten(Cake) are not mutex



Heuristics from Planning Graph

The estimate cost of any goal literal is the first level at which it appears

- Estimate is admissible for individual literals
- Can be improved by serializing the graph to allow only one action per level

Heuristics from Planning Graph

The estimate cost of a conjunction of goal literals: 3 heuristics

- Max-level
  - The largest level of a literal in the conjunction
  - Admissible, not necessarily accurate
- Level sum
  - Under subgoal independence assumption, add the level costs of goals
  - Inadmissible, works well for largely decomposable problems
- Set level
  - Find level at which all literals appear w/o any pair of them being mutex
  - Works extremely well when there is interaction among subplans

Plan from Planning Graph

The GraphPlan algorithm expands the graph with new levels until there are no mutex links between the goals.

To extract the actual plan, the algorithm searches backwards in the graph.

The plan extraction is the difficult part and is usually done with greedy-like heuristics.

Where we are

So far we have talked about:

- Problem Solving/Search
- Reasoning / Planning

Coming up next: the tools that got us out of the AI winter

- Uncertainty
- Learning

Where we are

1970-90: Knowledge-based approaches

- 1969-79: Early development of knowledge-based systems
- 1980-88: Expert systems industry booms
- 1988-93: Expert systems industry busts: "AI Winter"

1990-: Statistical approaches

- Resurgence of probability, focus on uncertainty
- General increase in technical depth
- Agents and learning systems… "AI Spring"?

Uncertainty?

So far we have considered environments that are:

- Fully observable
- Deterministic

Uncertainty comes into play when the environment is:

- Partially observable. Partial observability may be due to:
  - Incomplete knowledge
  - Faulty sensors/limited sensors
- Non-deterministic:
  - actions do not always go as planned – swimming in the ocean (waves and rip currents)
  - Faulty actuators