Sequential Decision-Making:
Automated Search Control

Siddharth Srivastava, Ph.D.
Assistant Professor
Arizona State University

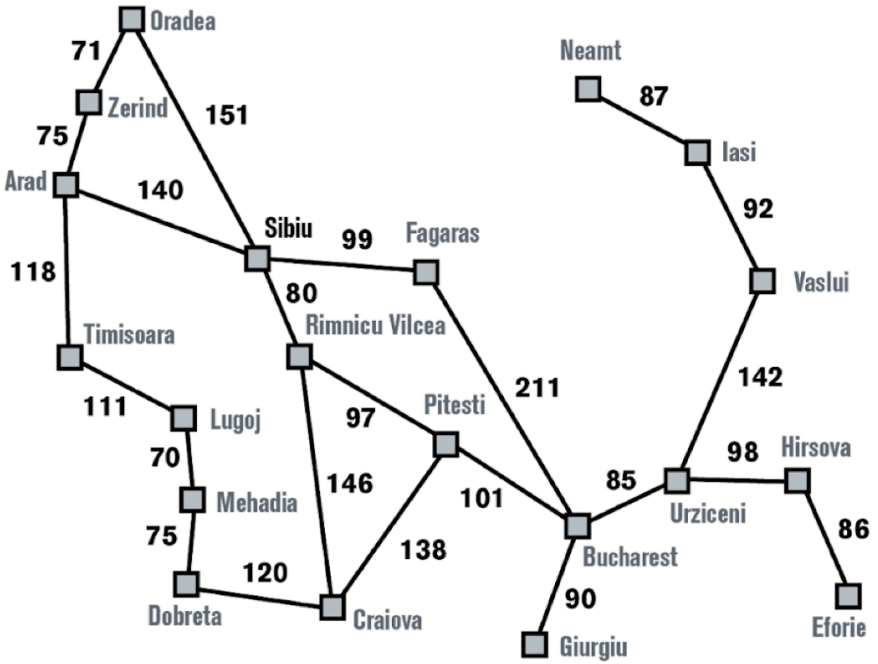
Ira A. Fulton Schools of Engineering
Arizona State University

Automated Search Control

Recall: PDDL Domain + Problem specification defines a search problem

Given:

- Graph $\langle V, E, A, \ell \rangle$
  - V set of vertices
  - E set of directed edges
  - A set of actions
  - $\ell: E \rightarrow \langle A, \mathbb{R}^+ \rangle$
- $v_i \in V$ start vertex
- G goal vertex set



Find a path from v_i to G

Automated Search Control

Relational representations make it easier to specify real-world planning problems

This lecture: solving planning problems under the assumption that the environment is

- Deterministic, fully observable

Most popular solution approach:

- Informed search without hand-coded heuristics
  - Pseudo-informed search

Heuristics and pruning strategies are derived automatically

- Using domain independent algorithms

Ideas for Generating Heuristics and Search Control

- Goal Decomposition (not quite)
- Planning Graph
- Landmarks

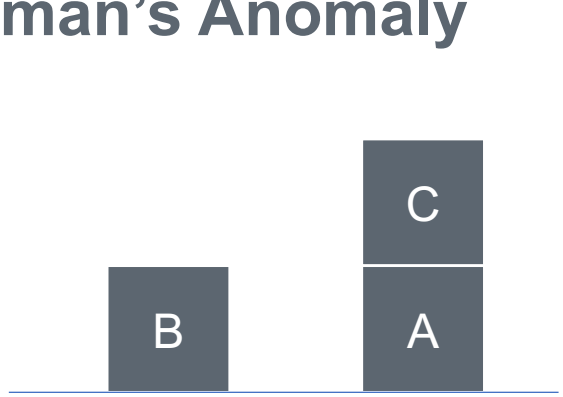
Goal Decomposition

Goal decomposition: Early approach motivated by the divide-and-conquer principle

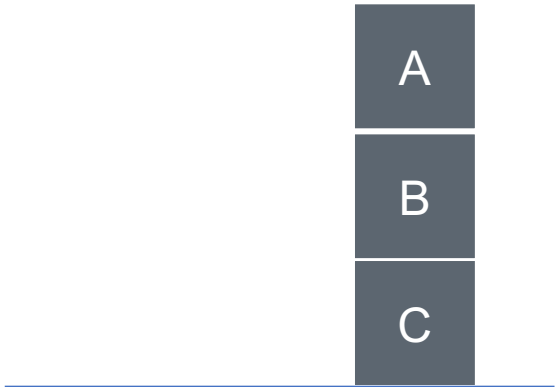
- If the goal requires multiple properties, compute plans achieving each independently
- Merge plans

What could go wrong?

Sussman's Anomaly



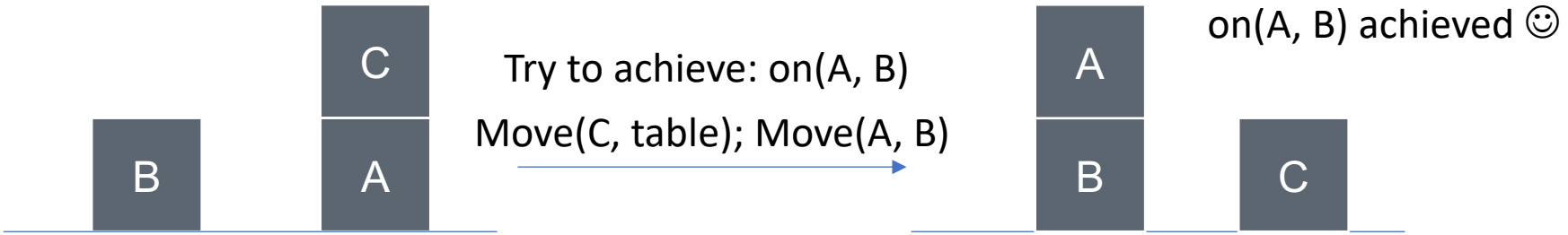
Initial state: on(A, B); on(B,C)



Goal: on(A, B) and on(B,C)

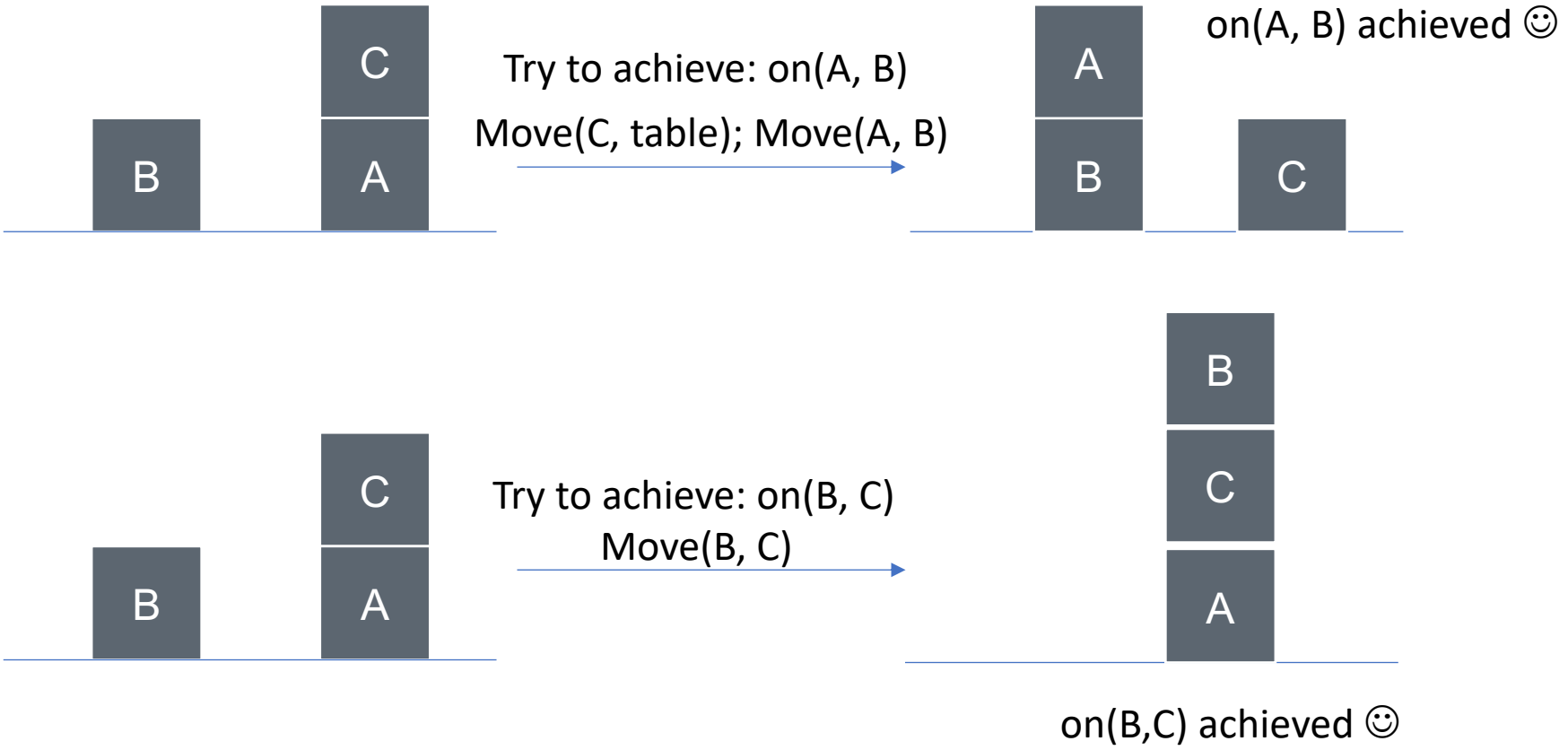
Goal Decomposition: Sussman's Anomaly

- Goal: on(A, B) and on(B,C)
- Idea: first find a plan for each conjunct; then merge them



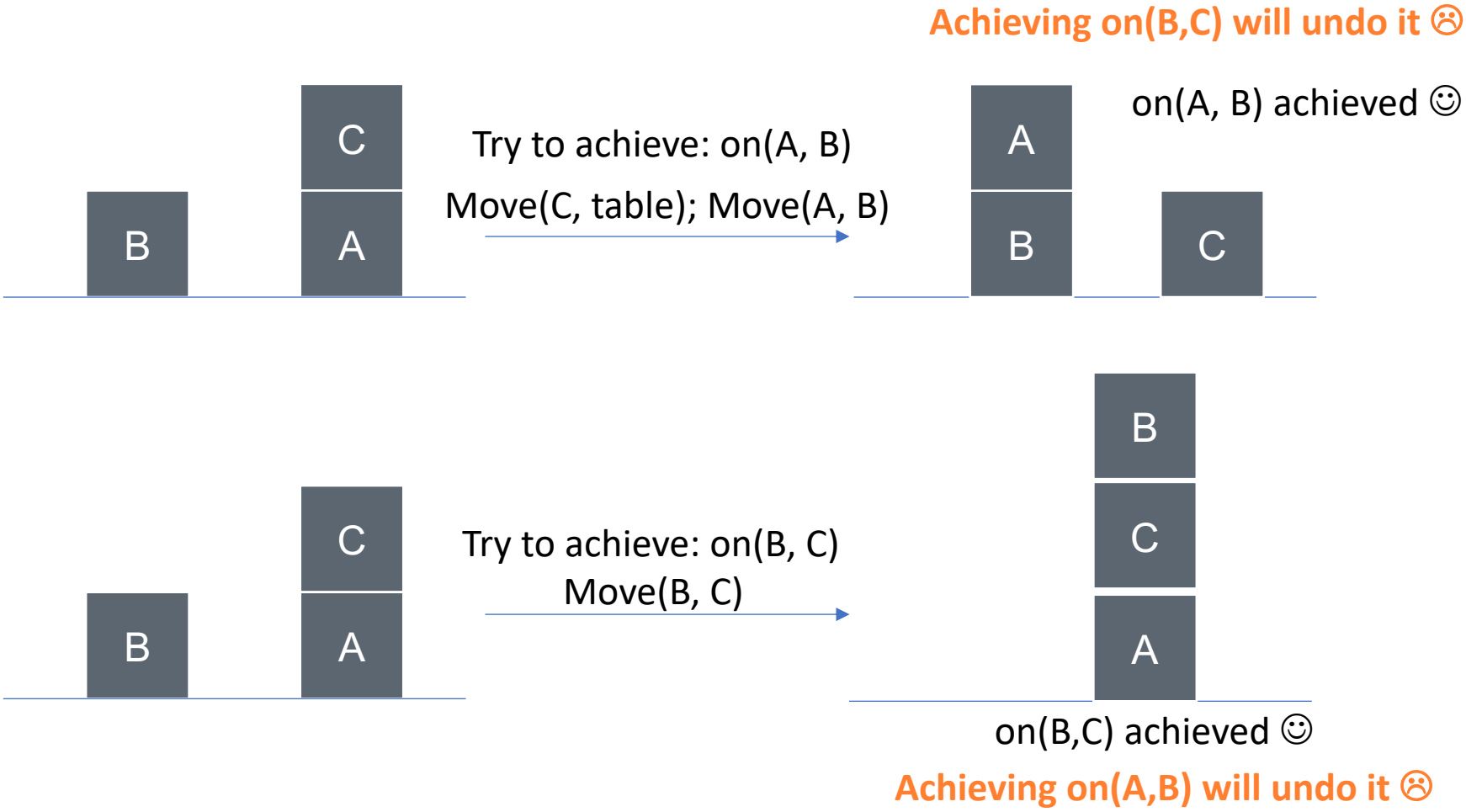
Goal Decomposition: Sussman's Anomaly

- **Goal: on(A, B) and on(B,C)**
- **Idea: first find a plan for each conjunct; then merge them**

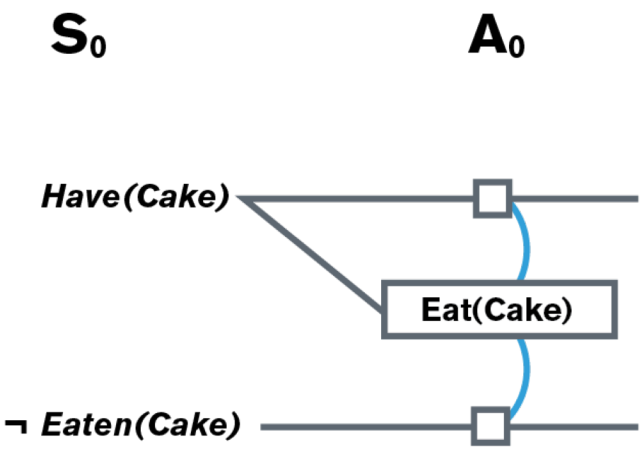


Goal Decomposition: Sussman's Anomaly

Problem: Merging these plans is difficult!



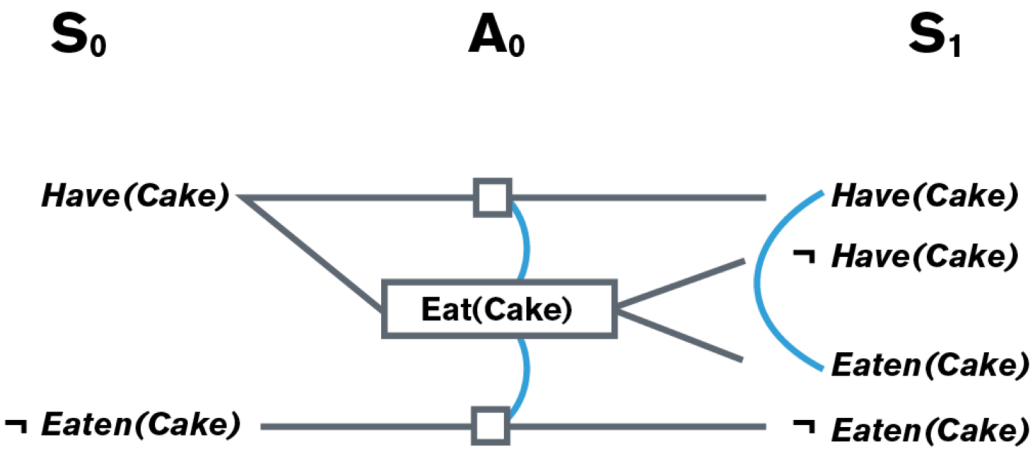
Planning Graph



$$
\begin{aligned}
&Init(Have(Cake))\\
&Goal(Have(Cake) \land Eaten(Cake))\\
&Action(Eat(Cake)\\
&\quad \text{PRECOND: } Have(Cake)\\
&\quad \text{EFFECT: } \neg Have(Cake) \land Eaten(Cake))\\
&Action(Bake(Cake)\\
&\quad \text{PRECOND: } \neg Have(Cake)\\
&\quad \text{EFFECT: } Have(Cake)
\end{aligned}
$$

[Russell and Norvig, 3rd Ed.]

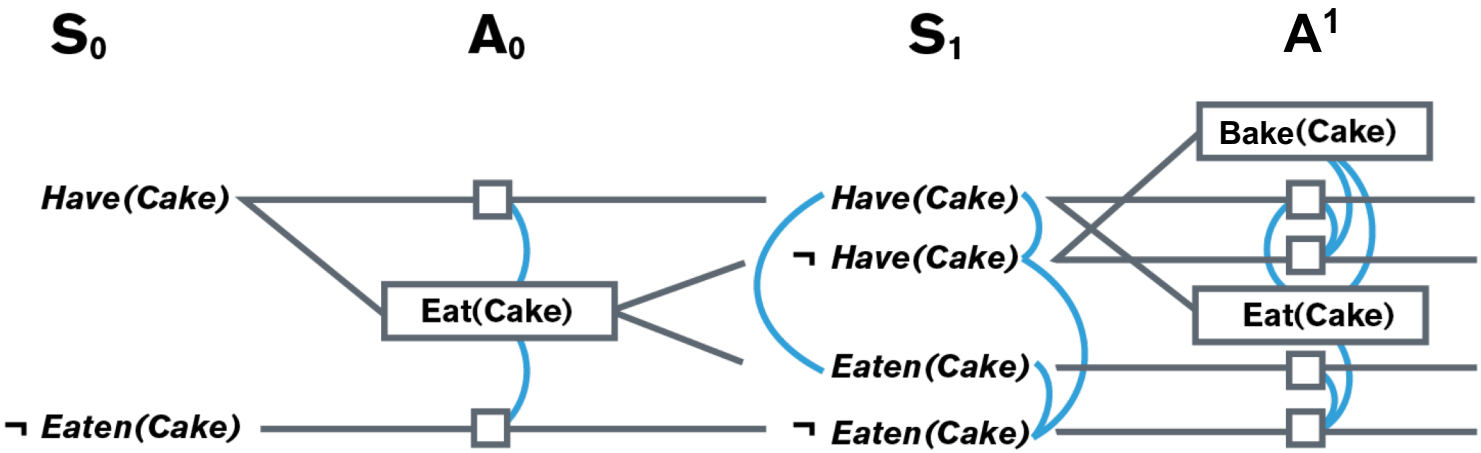
Planning Graph



$$
\begin{aligned}
&Init(Have(Cake))\\
&Goal(Have(Cake) \wedge Eaten(Cake))\\
&Action(Eat(Cake)\\
&\quad \text{PRECOND: } Have(Cake)\\
&\quad \text{EFFECT: } \neg\, Have(Cake) \wedge Eaten(Cake))\\
&Action(Bake(Cake)\\
&\quad \text{PRECOND: } \neg\, Have(Cake)\\
&\quad \text{EFFECT: } Have(Cake)
\end{aligned}
$$

[Russell and Norvig, 3rd Ed.]

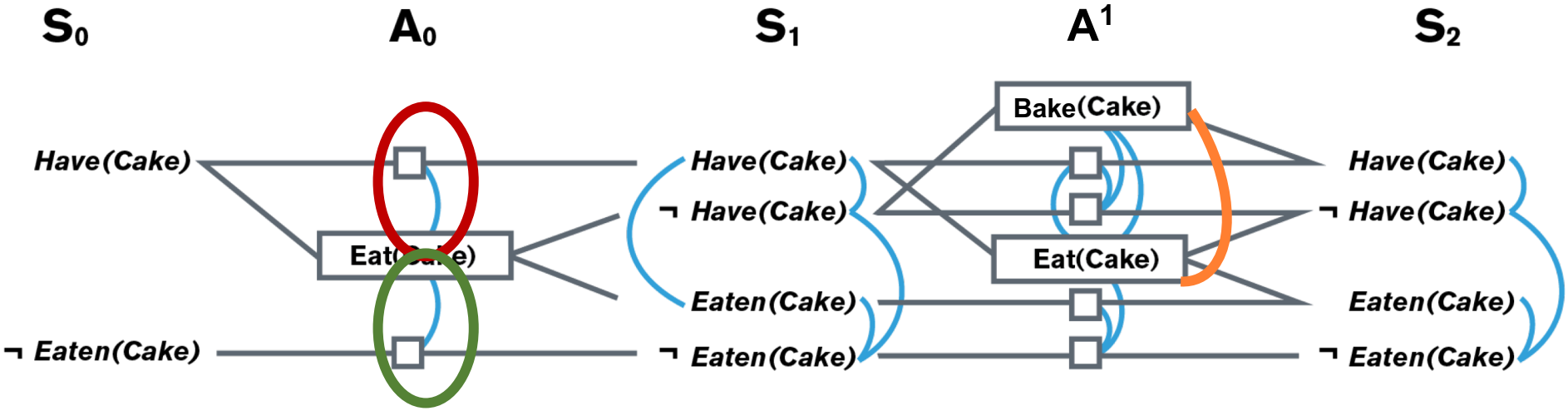
Planning Graph



$$Init(Have(Cake))$$
$$Goal(Have(Cake) \land Eaten(Cake))$$
$$Action(Eat(Cake)$$
$$\text{PRECOND: } Have(Cake)$$
$$\text{EFFECT: } \neg Have(Cake) \land Eaten(Cake))$$
$$Action(Bake(Cake)$$
$$\text{PRECOND: } \neg Have(Cake)$$
$$\text{EFFECT: } Have(Cake)$$

[Russell and Norvig, 3rd Ed.]

Planning Graph



Actions in a layer are *mutex* if:

Inconsistent Effects: one action's effect negates another's effect

Interference: one action negates another's precondition

Competing needs: one action requires negation of another's precondition

Extracting Plans From the Planning Graph

- **Mutex relations can be used to try to extract plans that achieve the goal**
- **It is possible that a plan is not feasible within a given expansion of graphplan**
  - Add another layer, try again
  - Until the graph levels off
  - If all the goal literals do not appear in the final level, goal is unreachable
- **Full graphplan algorithm: sound and complete**

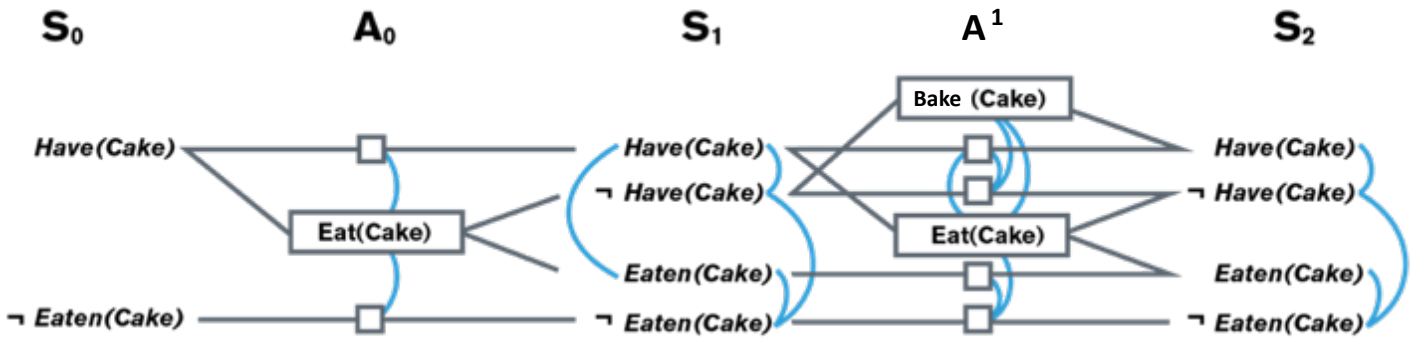
Extracting Heuristics from the Planning Graph

Expand the graph until you reach a fixed point

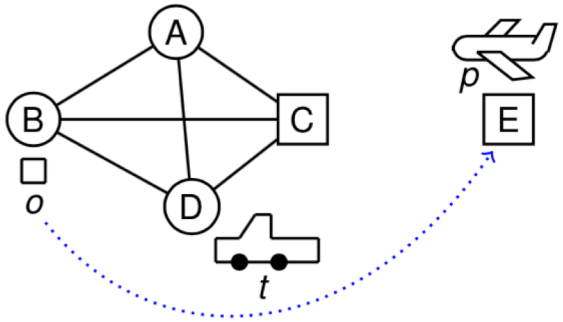
If a literal doesn't occur in the last layer, it cannot be achieved

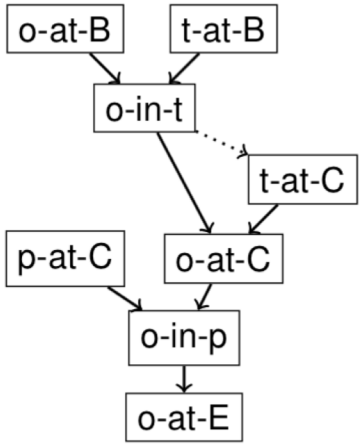
Level-cost heuristic

- Estimated cost of achieving a goal literal = level at which it first appears. Admissible?
- What about goals that include multiple literals?



Landmarks in Planning



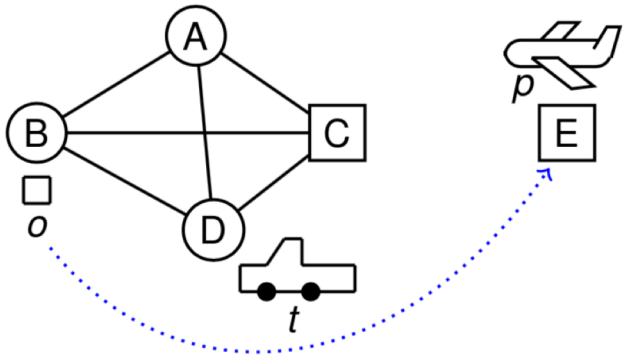


- A **landmark** is a formula that **must** become true at some point in every plan
- An **action landmark** is an action that **must** occur in every plan
- Landmarks and action landmarks can be ordered

Landmarks in Planning

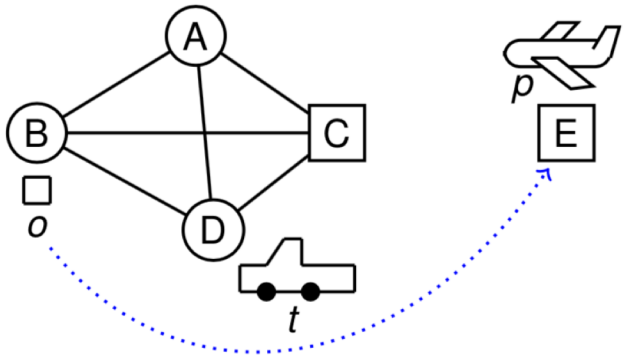
Backchaining process:

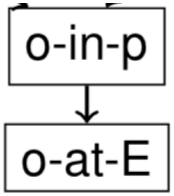
- Every goal is a landmark
- If B is a landmark and A is a common precondition in all actions that achieve B,
  - A is a landmark
  - $A \preccurlyeq B$



[Images credit: Sylvia Richter]

Landmarks in Planning

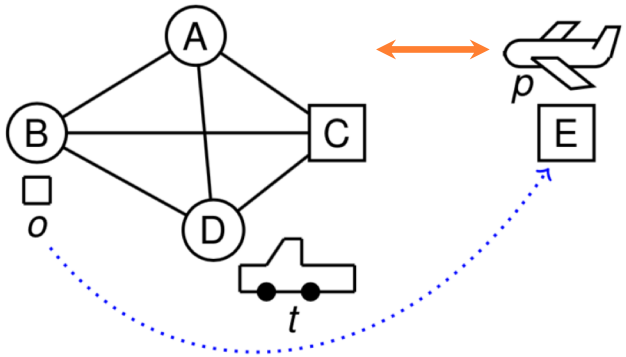


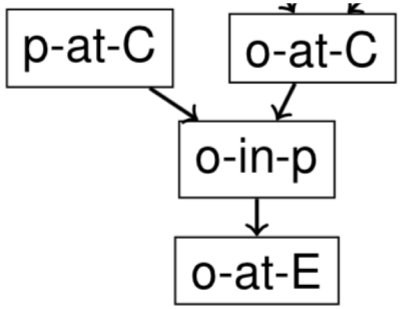


Backchaining process:

- Every goal is a landmark
- If B is a landmark and A is a common precondition in all actions that achieve B,
  - A is a landmark
  - $A \preccurlyeq B$

Landmarks in Planning

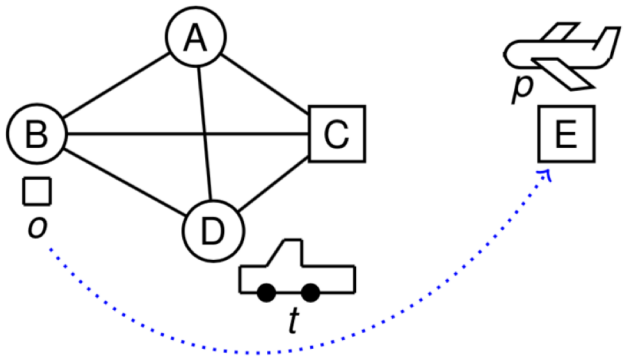


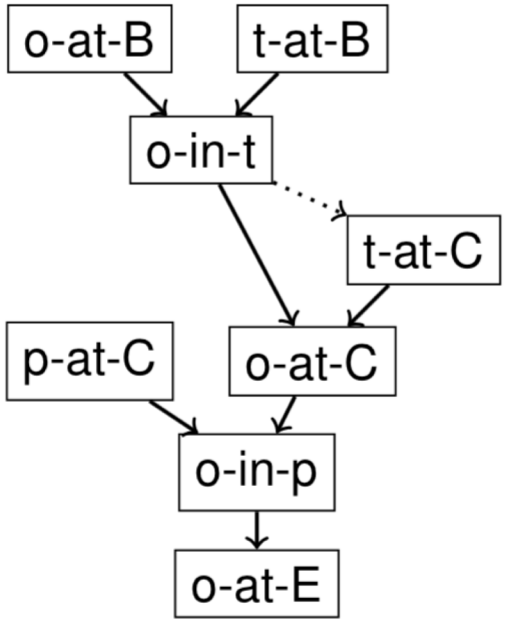


Backchaining process:

- Every goal is a landmark
- If B is a landmark and A is a common precondition in all actions that achieve B,
  - A is a landmark
  - $A \preccurlyeq B$

Landmarks in Planning





Backchaining process:

- Every goal is a landmark
- If B is a landmark and A is a common precondition in all actions that achieve B,
  - A is a landmark
  - $A \preccurlyeq B$

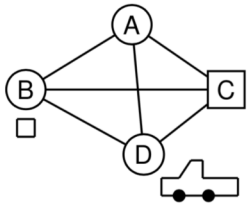
Using Landmarks as Heuristics

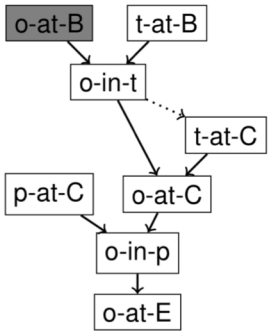
Two ways of using landmarks:

- subgoals,
- components in heuristic estimate

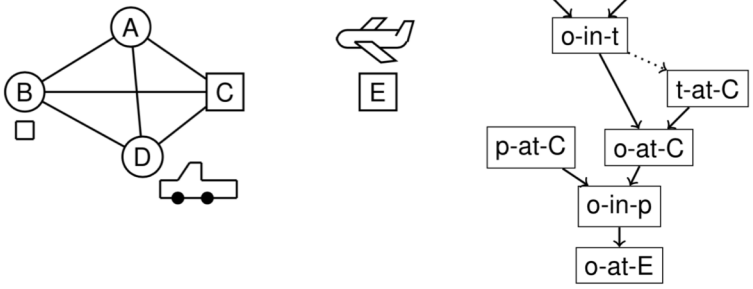
Subgoals:

- Select earliest landmark(s)
- Set as disjunctive goal for any planner
- After plan is found, repeat

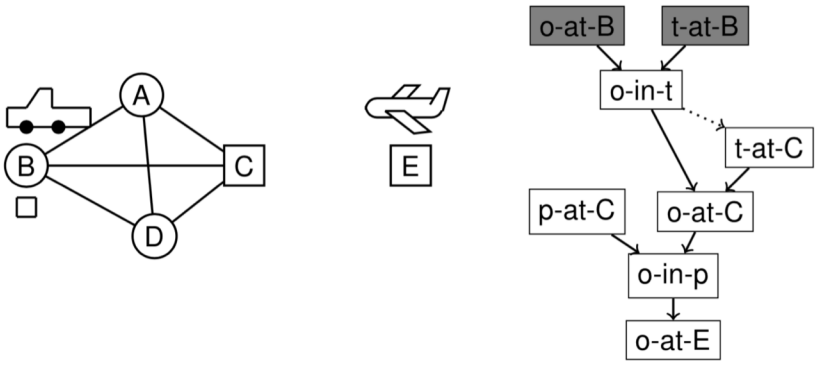




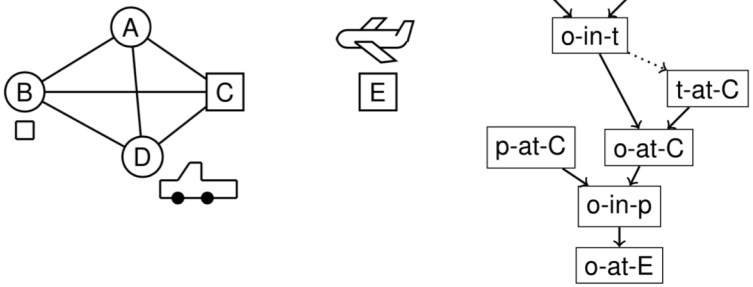
- Partial plan: $\emptyset$
- Goal: t-at-B $\vee$ p-at-C



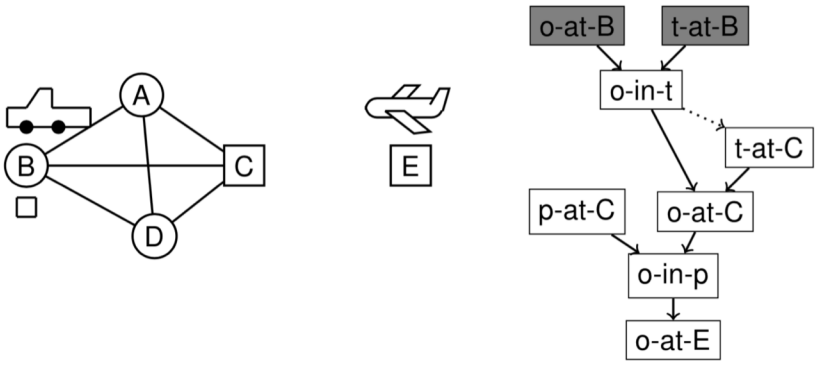
- Partial plan: $\emptyset$
- Goal: t-at-B $\vee$ p-at-C



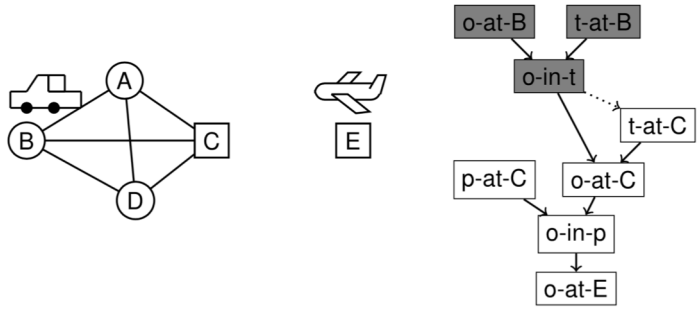
- Partial plan: Drive-t-B
- Goal: o-in-t $\vee$ p-at-C



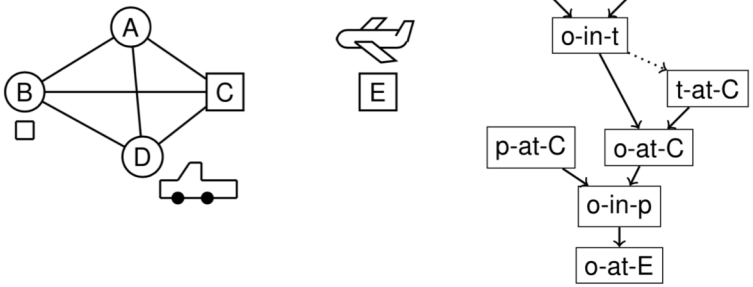
- Partial plan: $\emptyset$
- Goal: t-at-B $\vee$ p-at-C



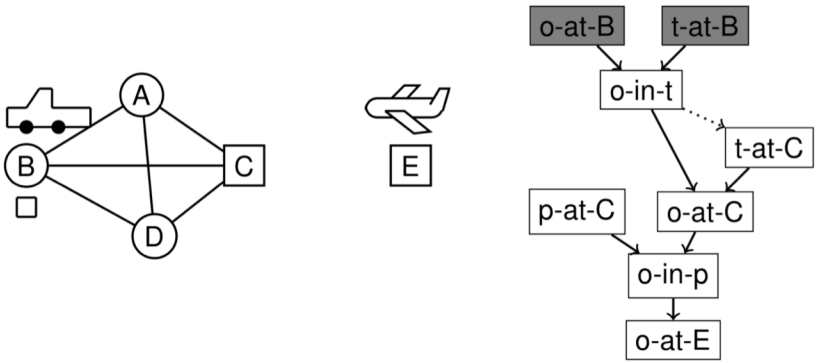
- Partial plan: Drive-t-B
- Goal: o-in-t $\vee$ p-at-C



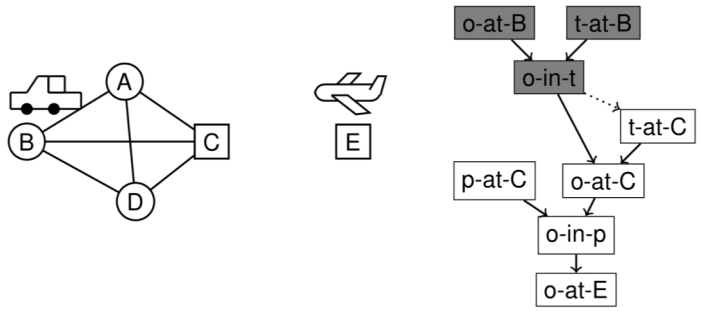
- Partial plan: Drive-t-B, Load-o-B
- Goal: t-at-C $\vee$ p-at-C



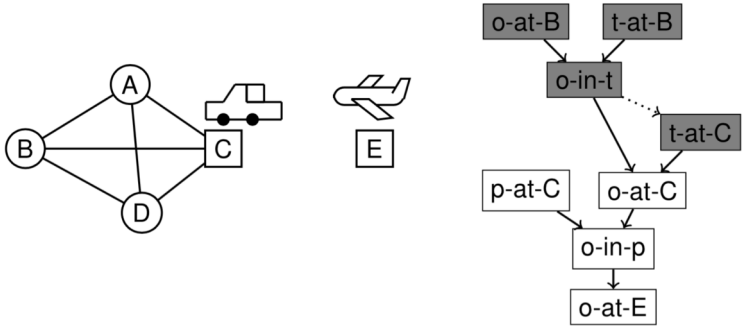
- Partial plan: $\emptyset$
- Goal: t-at-B $\vee$ p-at-C



- Partial plan: Drive-t-B
- Goal: o-in-t $\vee$ p-at-C



- Partial plan: Drive-t-B, Load-o-B
- Goal: t-at-C $\vee$ p-at-C



- Partial plan: Drive-t-B, Load-o-B, Drive-t-C
- Goal: o-at-C $\vee$ p-at-C

Using Landmarks

Using landmarks as subgoals is not guaranteed to be efficient

- Order of achieving mutually unordered landmarks can affect size of the plan
  - Variant of sussman's anomaly
- May fail for problems with dead-ends

Alternative:

- Use # unachieved landmarks as a part of the heuristic estimate