Linköping University

Automated Planning

Landmark Heuristics for State Space Planning

Jonas Kvarnström
Department of Computer and Information Science
Linköping University

jonas.kvarnstrom@liu.se – 2019

Landmarks (1)

jonkv@ida

Landmark:

"a **geographic feature** used by explorers and others to **find their way** back or through an area"

Landmarks (2)

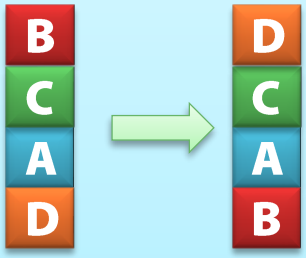
jonkv@ida

Landmarks in planning:
Something you must ***achieve*** *or* ***use*** in ***every solution*** to a problem instance

Assume we are considering a state *s*…

Fact Landmark for *s*:

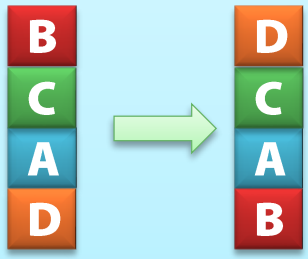
A **fact** that must be true
at some point
in *every* solution starting in *s*



clear(A)
holding(C)
…

Formula Landmark for *s*:

A **formula** that must be true
at some point
in *every* solution starting in *s*



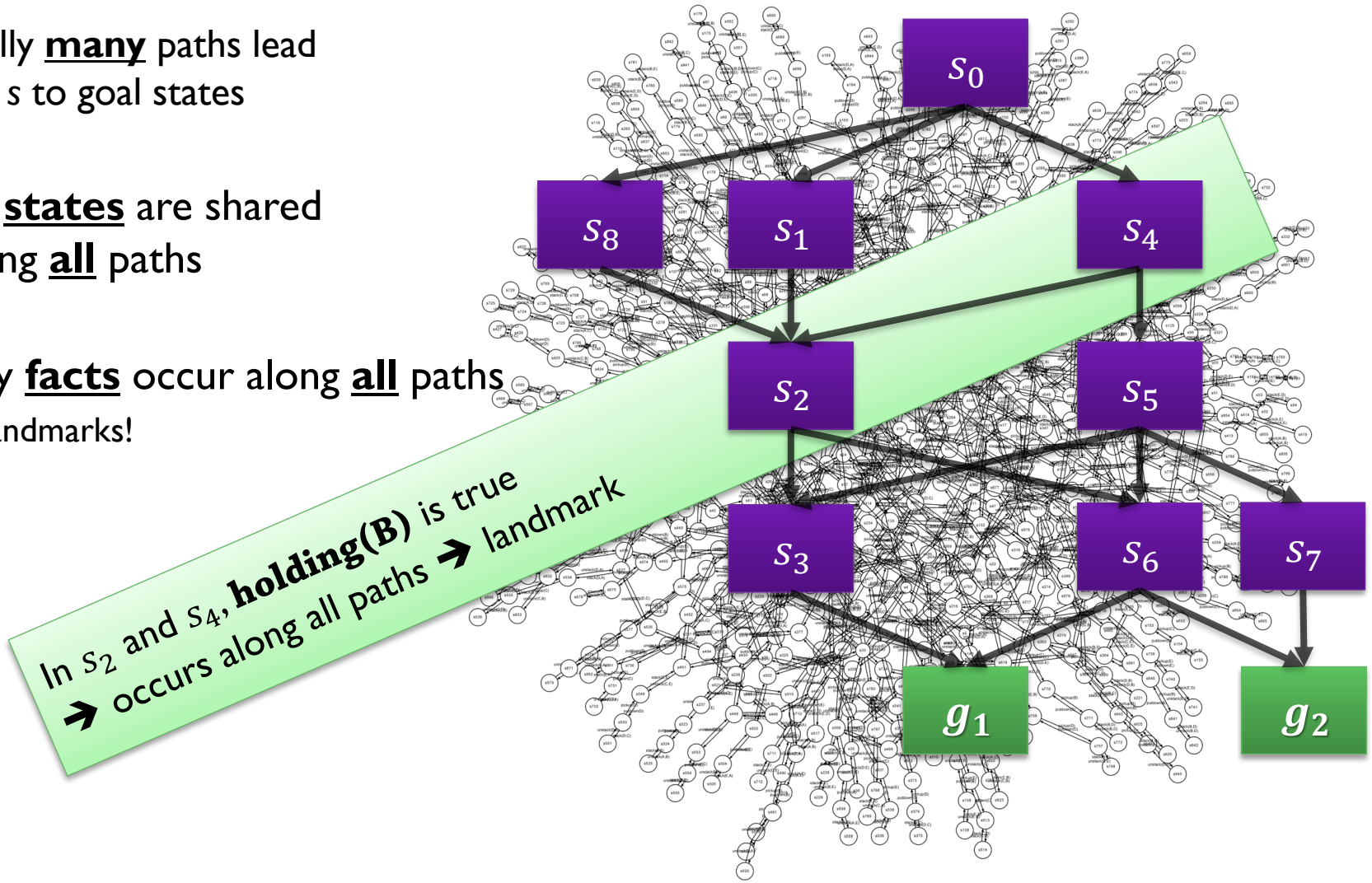
clear(A) ∧ handempty
…

Landmarks (3)

jonkv@ida

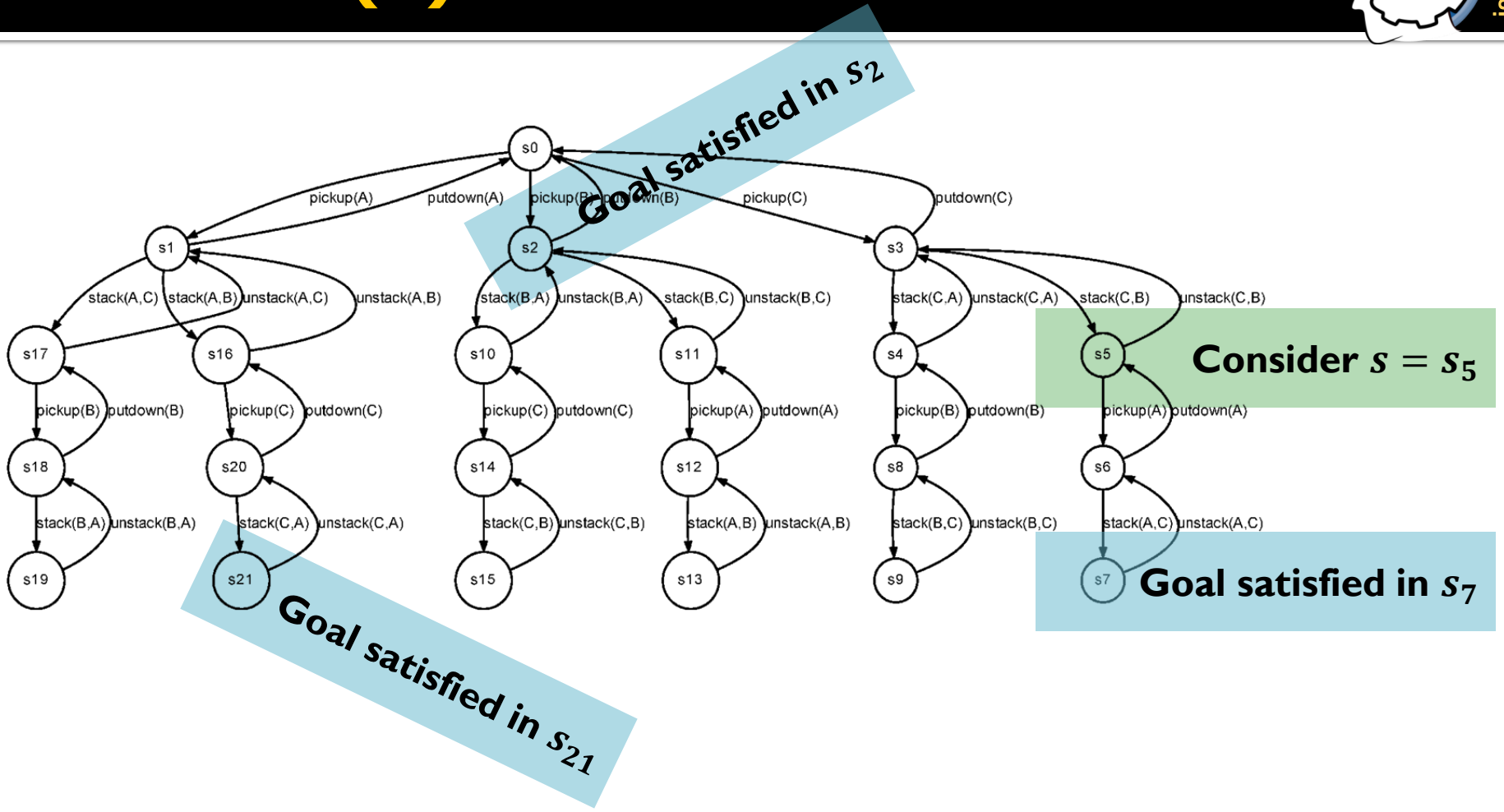
Facts and formulas, not states! Why?

- Usually **many** paths lead from s to goal states
- Few **states** are shared among **all** paths
- Many **facts** occur along **all** paths
  - Landmarks!



Landmarks (4)

jonkv@ida



Is there a landmark <u>state</u> s_{lm} we <u>must</u> pass to reach <u>some</u> goal from s_5?

No! But we may have to pass <u>different</u> states satisfying the same <u>facts</u> f_{lm}!

Landmarks (5): Misunderstandings

jonkv@ida

Not ”we must reach (pass through) **the** landmark state”!

Instead ”we must reach ***some state*** that satisfies the fact/formula landmark”

Not ”A landmark fact is a state that…”

A fact is not a state.
A state *consists* of *many* facts.

(”A word is a sentence that…”)

A landmark fact is **not** ”a fact that is true in every solution”

A solution is a plan.
Facts are true in *states*.

A landmark fact is ”a fact that is true in some state along every path from the initial state to any goal state”.

Can you be ”close” to a landmark?

You ***can*** be in a state s that is close to ***another*** state s' satisfying the landmark.

Problem: How to know?

Distance is ”number of edges” or ”cost of reaching”, not $\Delta x/\Delta y$. And the graph may not even be expanded yet.

Landmarks (6)

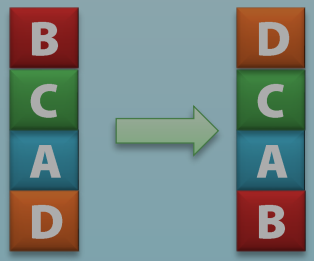
jonkv@ida

Landmarks in planning:
Something you must *pass by/through* in *every solution* to a specific planning problem

Assume we are currently in state *s*…

Fact Landmark for *s*:

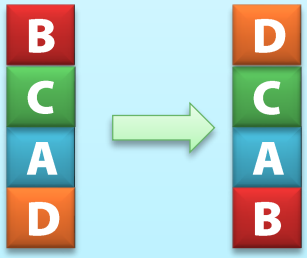
A **fact** that must be true
at some point
in *every* solution starting in *s*



clear(A)
holding(C)
…

Action Landmark for *s*:

An **action** that must be used
in *every* solution starting in *s*



unstack(B,C)
putdown(B)
stack(D,C)

…so the effects of action landmarks are *fact landmarks*, and so are their *preconds*

(except those facts that are already true in *s*)

Landmarks (7)

jonkv@ida

- Generalization:
  - **Disjunctive** action landmark $\{a_1, a_2, a_3\}$ for state s
    - Every solution starting in state s and reaching a goal must use *at least one* of these actions

Finding Landmarks:
A (Too) General Technique

Finding Landmarks: General Technique

jonkv@ida

- One general technique for **discovering landmarks**:

Current planning problem, P

Initial state does not include atom A

Modified planning problem, P'

Removed all actions that add atom A

If this problem (P') is **unsolvable**…

Test:
Delete relaxation of P' is unsolvable,
or $h_m(s_0) = \infty$, or …
➔ P' is unsolvable

…then **every** solution to P must use one of the removed actions

➔ **Action set is a disj. act. landmark**
➔ **Atom A is a fact landmark**

Unsolvable when removing a set of actions
➔ some action in the set must be used ➔ disjunctive action landmark!

Finding Landmarks: General Technique (2)

jonkv@ida

- This technique is very general
  - Applicable to *any* planning problem, *any* atom

- General techniques tend to be **widely applicable** but **slow**…

Verifying Landmarks (1)

jonkv@ida

- How difficult is it to **verify** that an action is an **action landmark**, in the **general** case?
  - Suppose we **can** verify this
  - Then given any STRIPS problem *P*, we can **determine if it has a solution**:
    - Add a new action:
      - **cheat**

        :precond true

        :effects goal-formula
    - If **cheat** is an action landmark, then it is *needed* in order to solve the problem
      ➔ the original problem was *unsolvable*
  - ➔ As difficult as solving the planning problem (PSPACE-complete)

Porteous et al (2001): On the Extraction, Ordering, and Usage of Landmarks in Planning

Verifying Landmarks (2)

jonkv@ida

- How difficult is it to **verify** that a fact is a **fact landmark**, in the **general** case?
  - Suppose we **can** verify this
  - Then given any STRIPS problem *P*, we can **determine if it has a solution**:
    - Add a new fact:
      - **cheated** (false in the initial state)
    - Add new action:
      - **cheat**
        :precond true
        :effects (and **cheated** goal-formula)
    - If **cheated** is a fact landmark,
      then **cheat** was necessary ➔ the original problem was unsolvable
  - ➔ Again , as difficult as solving the planning problem

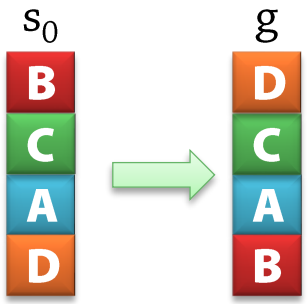
But of course there are special cases…

Finding Landmarks: Efficiently

Means-Ends Analysis

jonkv@ida

- Discover landmarks using **means-ends analysis**



Unachieved goal facts
are (obviously) fact landmarks:
clear(D), on(D,C), on(A,B), ontable(B)

fact-landmarks ← g – s

Suppose on(D,C) is a landmark,
on(D,C) is not true in the current state (s)
➔ we must *cause* **on(D,C)** with an action
➔ compute *achievers* = **{ stack(D,C) }**

do {
  for each p in fact-landmarks {
    // Create *disjunctive* action landmark
    achievers ← $\{a \in A \mid p \in \text{eff}(a)\}$

All achievers require these candidates =
{ ***holding(D), handempty, clear(C), …*** }

candidates ← $\bigcap_{a \in achievers} pre(a)$

handempty is already true, but
new = { **holding(D), clear(C)** } are *not*

new ← candidates – s
fact-landmarks ← fact-landmarks ∪ new

Maybe we can find more landmarks
related to achiving *those*!

  }
} **until** no more fact-landmarks found

Actions, Forward

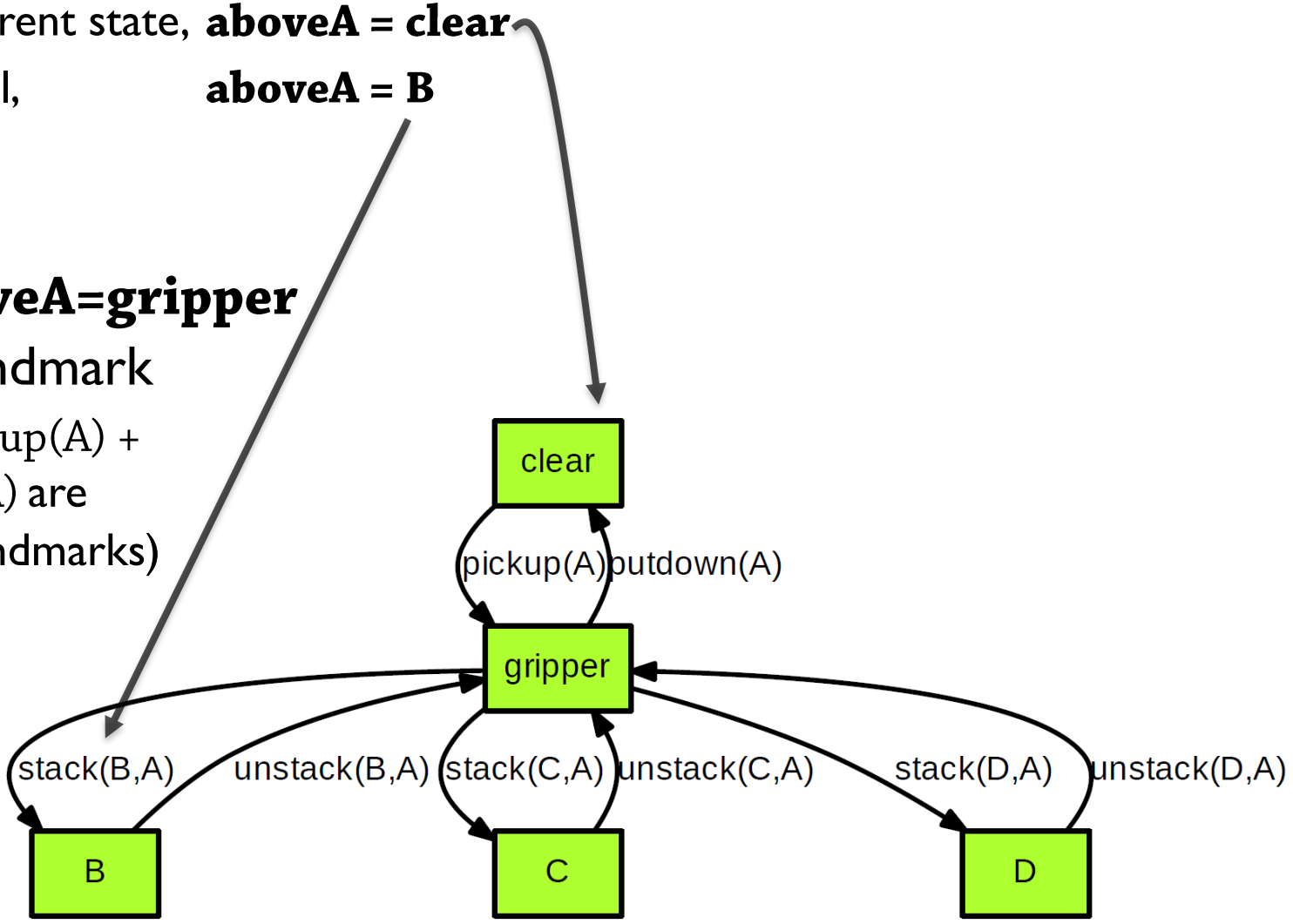
jonkv@ida

- Extensions to backwards means-ends analysis:
  - **Effects of disjunctive action landmarks**:
    - All shared **effects** must also take place regardless of the ”chosen” action, similarly to shared *preconditions* on the previous page
    - Given a disjunctive action landmark,
      every fact in $(\bigcap\{\text{eff}(a)|a \in \text{landmark}\} - s)$ is a fact landmark for s

Landmarks from DTGs

jonkv@ida

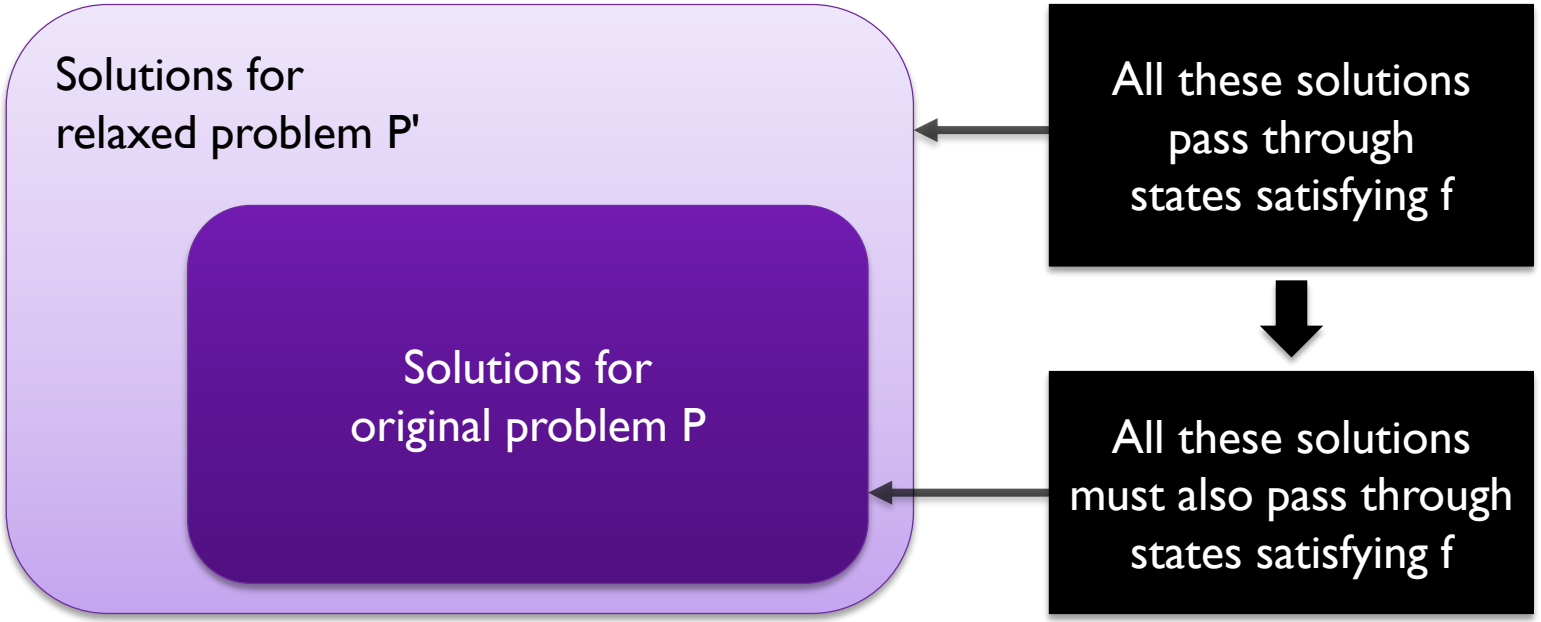
- Another method: Use **domain transition graphs**:
  - In the current state, **aboveA = clear**
  - In the goal, **aboveA = B**
- Then **aboveA=gripper** is a fact landmark
  - (And pickup(A) + stack(B,A) are action landmarks)



Landmarks and Relaxation

jonkv@ida

- Assume a problem P, and a **relaxed problem** P'
  - Suppose f is a fact landmark for P'



  - Then f is a fact landmark for the original problem as well!
  - Similarly for action landmarks, etc.

Landmarks

jonkv@ida

- Many other techniques exist…
  - Beyond the scope of the course

Landmark Ordering

Landmark Ordering (1)

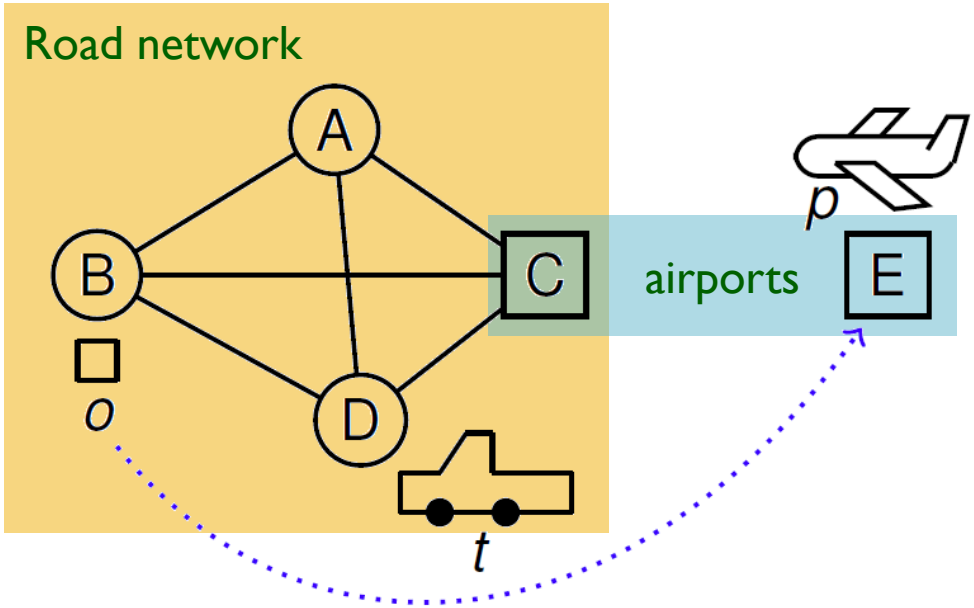
jonkv@ida

- Sometimes we can find or approximate **necessary orderings**
  - We must achieve holding(A), *then* holding(B)

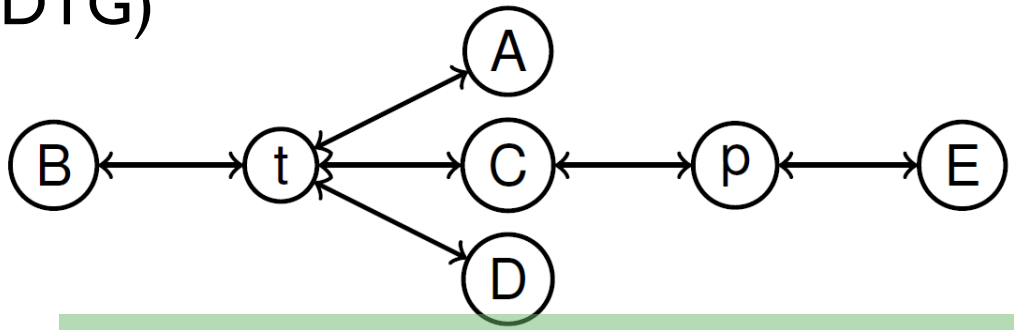
Landmark Ordering (2): Example Problem

jonkv@ida

- Example Problem:
  - Truck t transports object o within road network A/B/C/D
  - Airplane p transports object between airports C/E
  - Goal: Object at E



- Domain transition graph (DTG) for **location-of-object**:

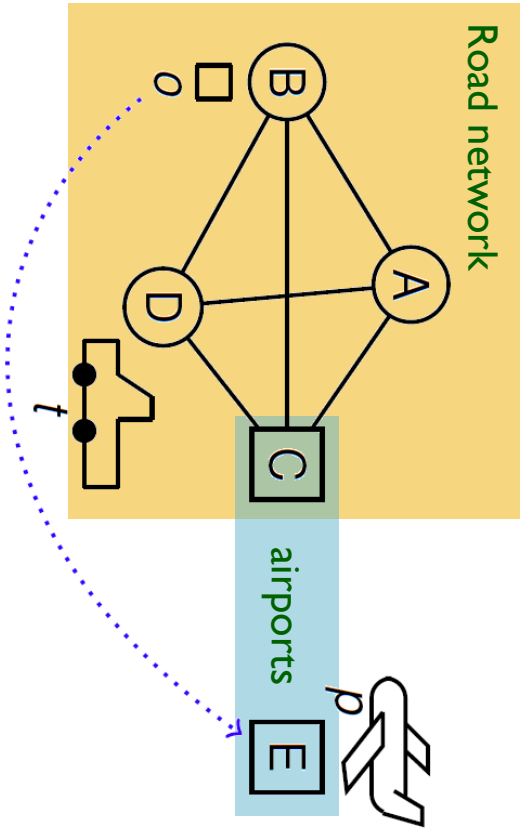


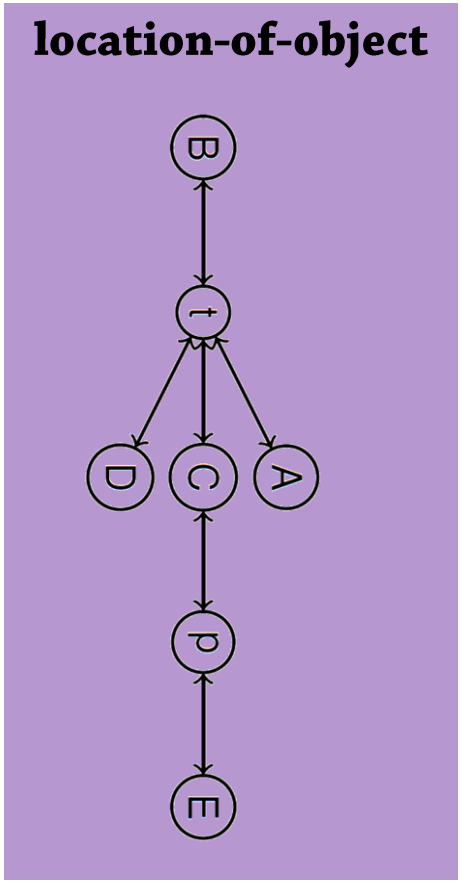
Note: Every **edge** in the road network corresponds to a **path** through **t** in the DTG!

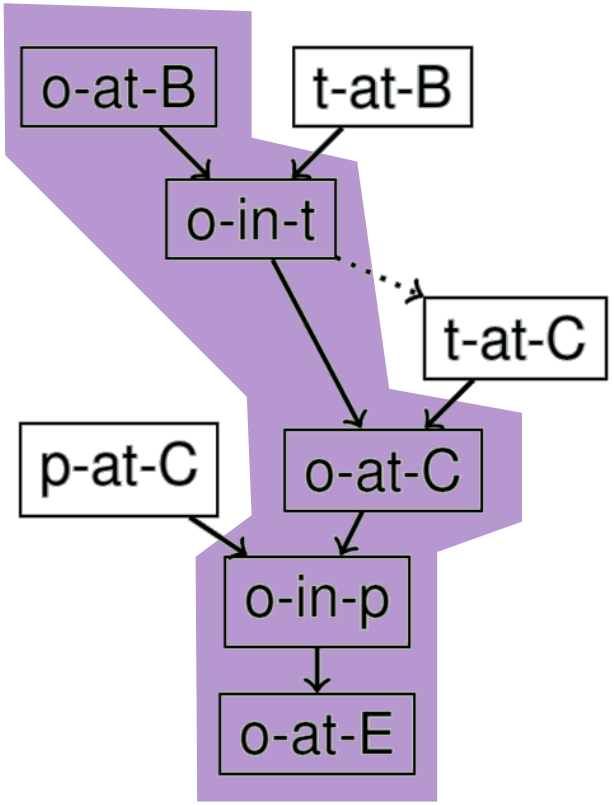
Landmark Ordering (3): Inference

jonkv@ida

- One way of inferring the order of landmarks:
  - Directly from the DTG!





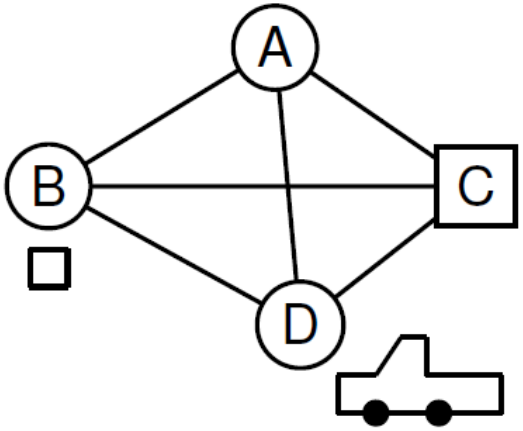


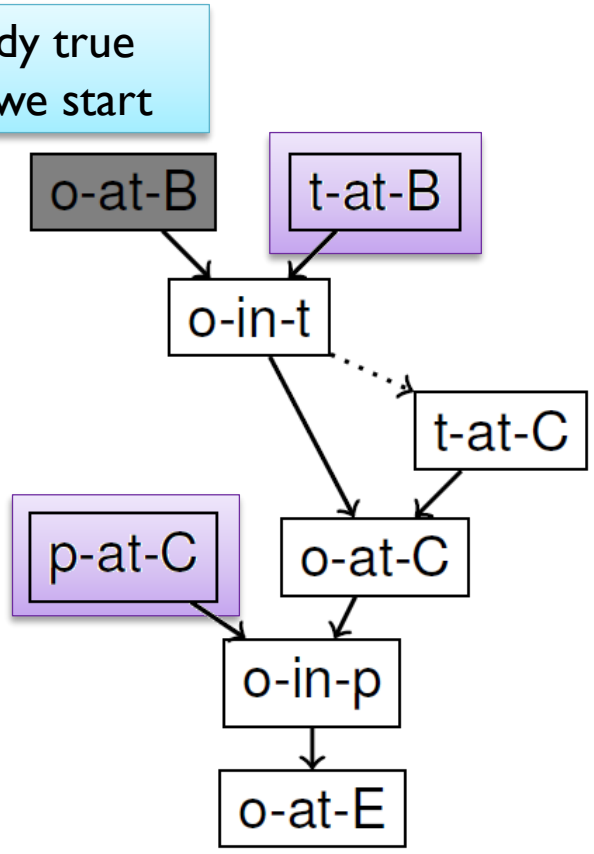
Using Landmarks:
As Subgoals

Landmarks as Subgoals (1)

jonkv@ida

- One use of **ordered** landmarks:
  - As **subgoals**: Try to plan for each landmark **separately** in the inferred **order**



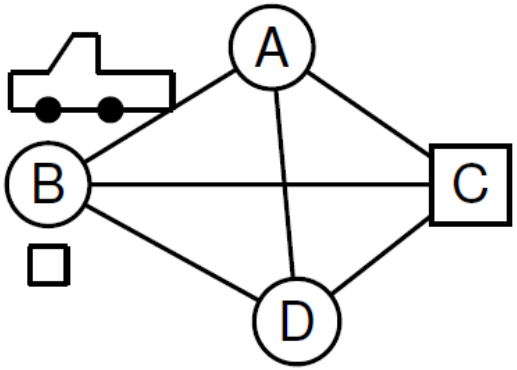


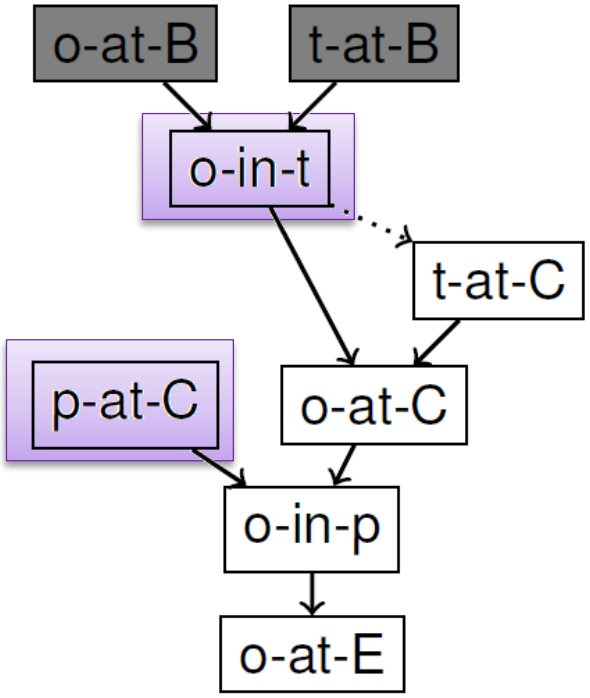
Two landmarks could be "first" (all predecessors achieved)
Current goal: t-at-B ∨ p-at-C (disjunctive!)

Landmarks as Subgoals (2)

jonkv@ida

Suppose we begin by achieving t-at-B:
Simple planning problem,
results in a single action -- drive(t, B)



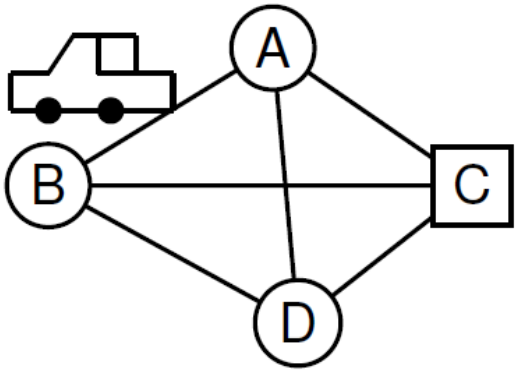


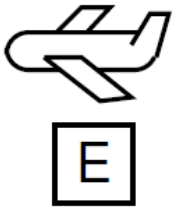
Current goal: o-in-T *or* p-at-C

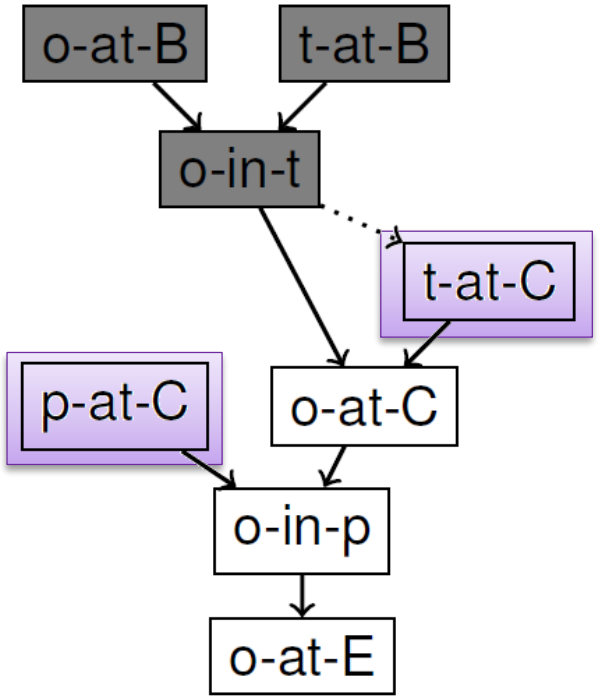
Landmarks as Subgoals (3)

jonkv@ida

Suppose we continue by achieving o-in-T:
Simple planning problem,
results in a single action -- load-truck(o,t,B)



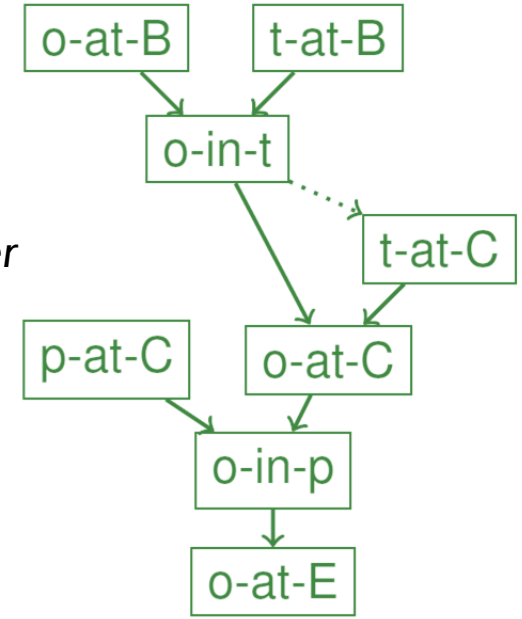




Landmarks as Subgoals (4)

jonkv@ida

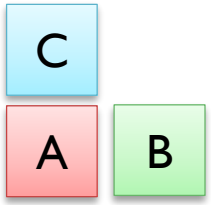
- Sometimes very helpful, but:
  - There are still ***choices*** to be made – backtrack points!
  - Optimizing for one **part** of the overall goal at a time:
    - Can't see the whole picture
    - Can miss opportunities:
      Cheapest solution *here* ➔ **more expensive** solution *later*
    - Can be incomplete:
      Cheapest solution *here* ➔ **impossible** to solve *later*



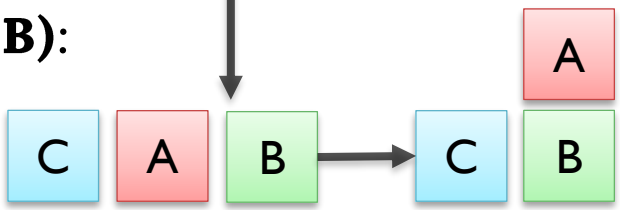
The Problem with Separating Subgoals

jonkv@ida

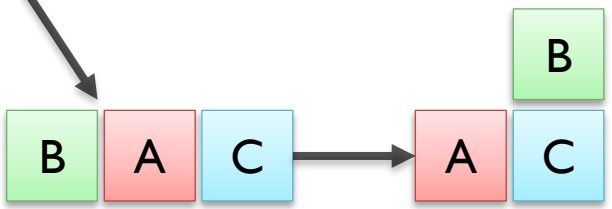
- The Sussman Anomaly (Gerald Sussman)
  - **Goal** is **on(A,B), on(B,C)**
  - **Now**:



  - **Idea: Achieve one at a time**
    - First, plan only for **on(A,B)**
    - Then, plan only for **on(B,C)**
  - Achieve **first** subgoal, **on(A,B)**:
    - unstack(C,A); putdown(C); pickup(A); stack(A,B)



  - Achieve **second** subgoal, **on(B,C)**:
    - unstack(A,B); putdown(A); pickup(B); stack(B,C) ➔ original goal destroyed!

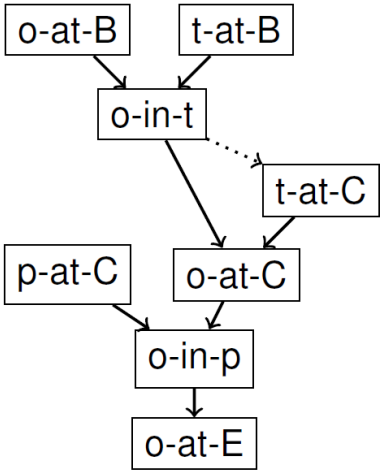


Using Landmarks:
To Define Heuristic Functions

Landmark Heuristics (1)

jonkv@ida

- The LAMA state space planner **counts** landmarks:
  - Landmarks that need to be achieved
    after reaching state s through path (action sequence) π



- **L(s,π) =**

(L \ Accepted(s,π))

All discovered landmarks,
minus those that are
accepted as achieved
(have become true *after*
predecessors are achieved!)

∪

ReqAgain(s,π)

Plus those we can show will
have to be *re-achieved*

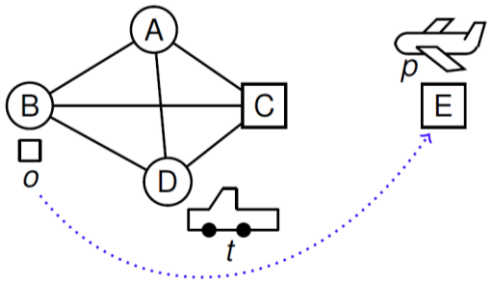
(Example:
Landmarks that were reached,
are no longer true,
but are required by the goal)

- $h(s) = |L(s, \pi)|$
- Not admissible: One action may achieve multiple landmarks!

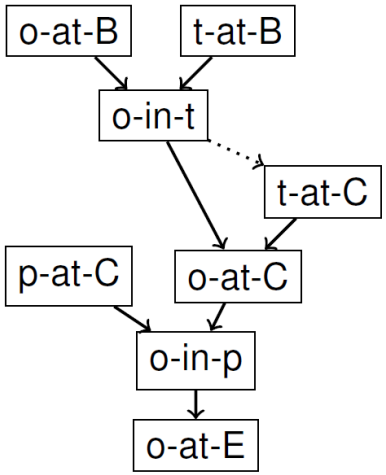
Landmark Heuristics (2)

jonkv@ida

- To achieve **admissible** heuristic estimates:
  - Idea: The cost of each action is *divided* across the landmarks it achieves
  - **Simplified example**:
    - Suppose there is a goto-and-pickup action of cost 10, that achieves both t-at-B and o-in-t
    - Suppose *no other action* can achieve these landmarks
    - One can then let (for example) cost(t-at-B)=3 and cost(o-in-t)=7



  - The sum of the cost of remaining landmarks is then an **admissible heuristic**
    - Must decide how to split costs across landmarks
    - Optimal split *can* be computed polynomially, but is still expensive

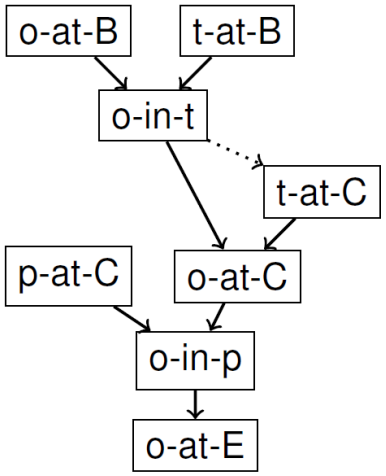


Using Landmarks:
For Problem Modification

Landmarks: Modified Problem

jonkv@ida

- Landmarks as a basis for a **modified planning problem**
  - Add new facts "achieved-landmark-*n*"
    - Concretely: *object-has-been-in-plane*
  - An action achieving a landmark makes the corresponding facts true
    - (load object plane) ➔ object-has-been-in-plane := true
  - The goal requires all such facts to be true
    - (**:goal** object-has-been-in-plane …)
  - ➔ Any *other* heuristic can be applied to the modified problem!
    - $h_{PDB}(s)$ – pattern databases: What is the cost of achieving object-has-been-in-plane?

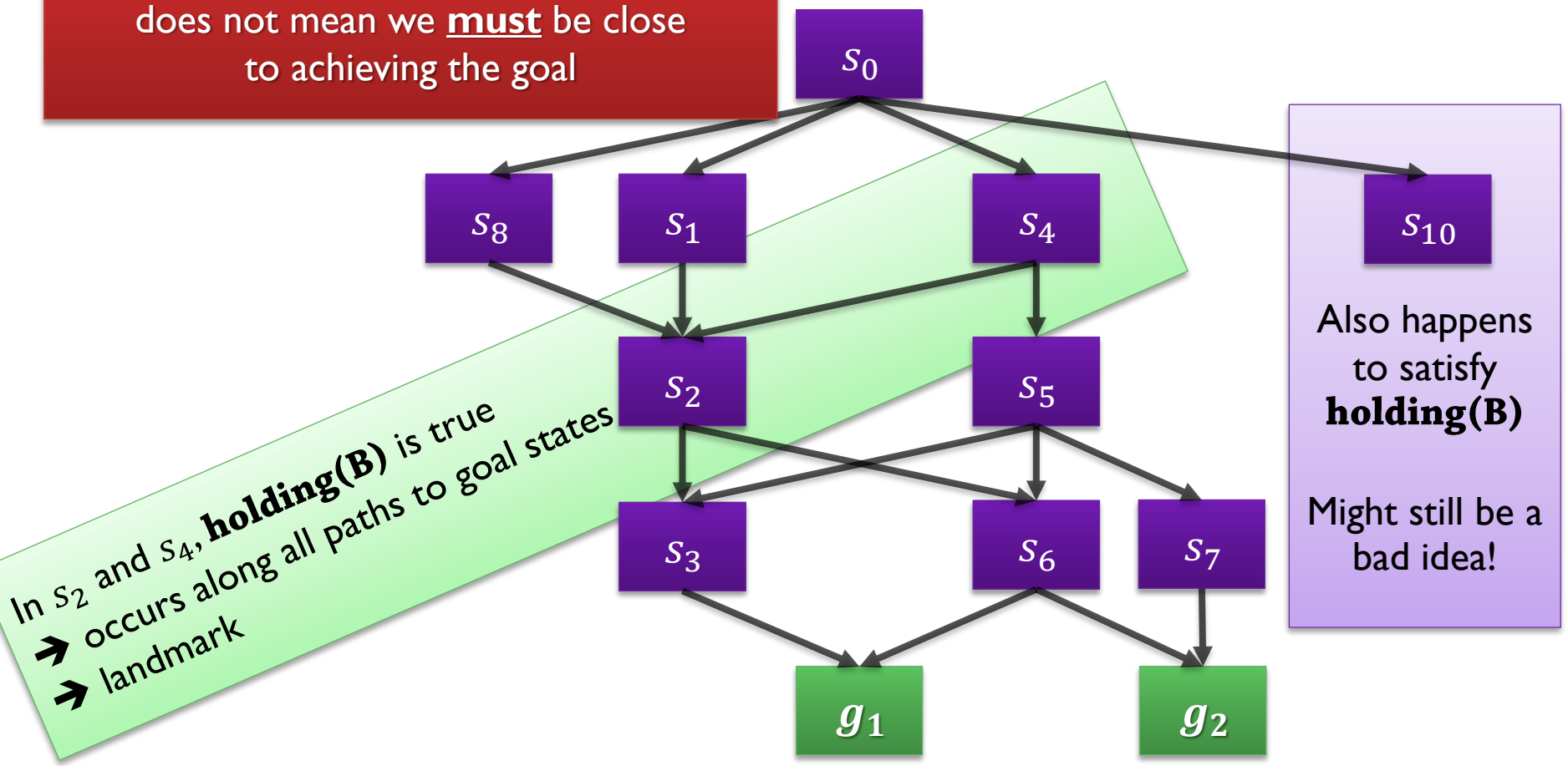


Landmarks: Ideas that Won't Work

Landmarks: More Misunderstandings

jonkv@ida

Satisfying a landmark
does not mean we **must** be close
to achieving the goal



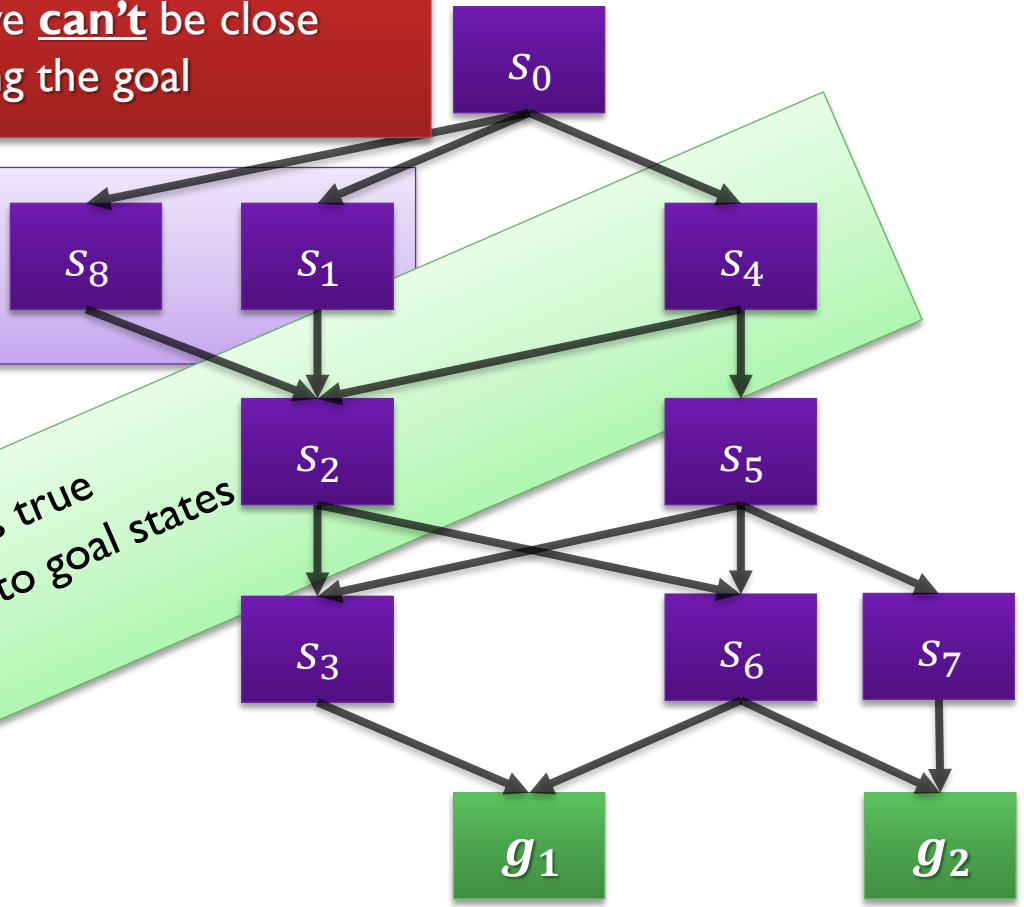
Landmarks: More Misunderstandings

jonkv@ida

Not satisfying a landmark
does not mean we **can't** be close
to achieving the goal

Satisfied no
landmarks yet?
Still a good idea

In s_2 and s_4, **holding(B)** is true
➔ occurs along all paths to goal states
➔ landmark

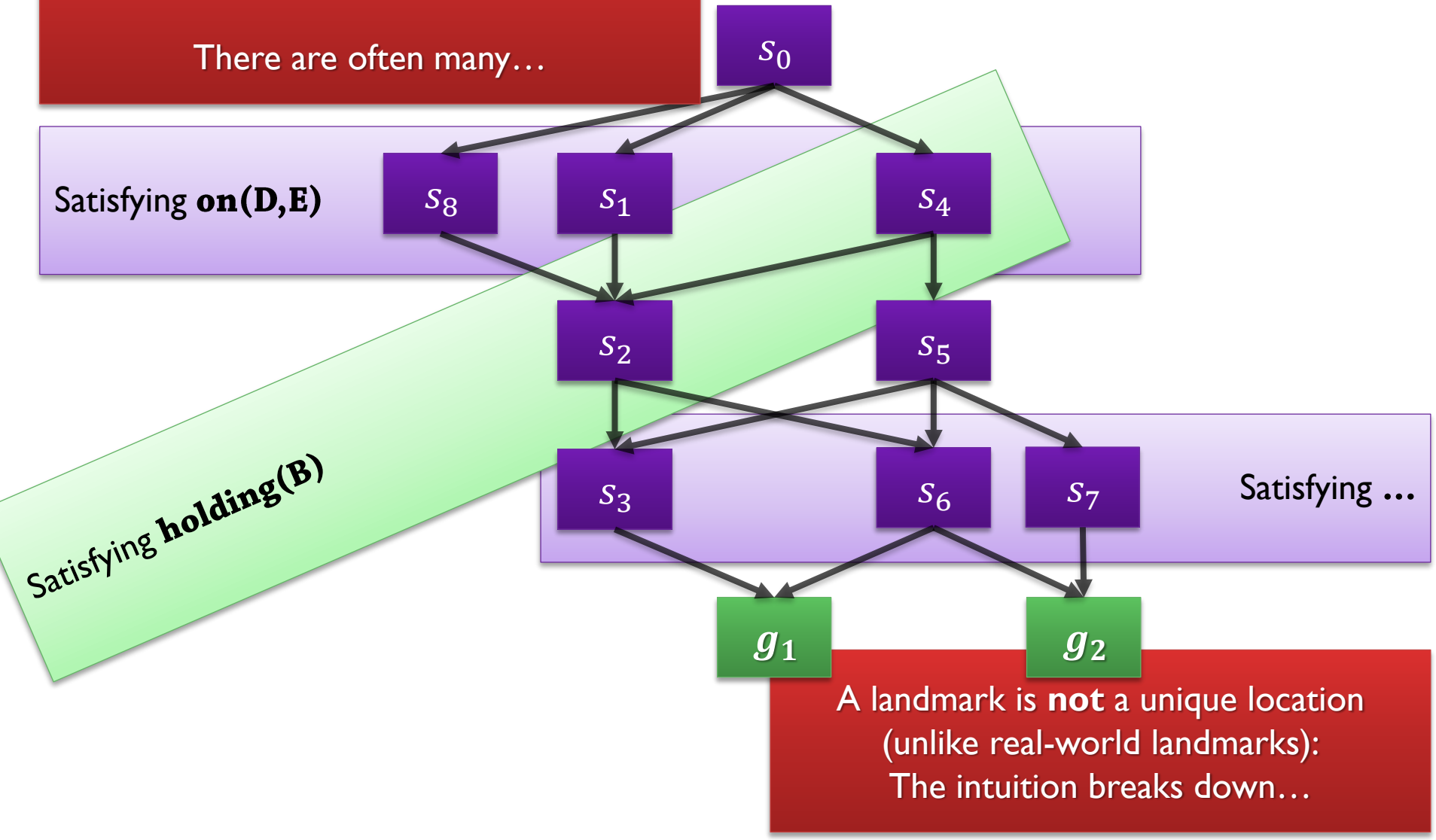


Landmarks: More Misunderstandings

jonkv@ida

We rarely talk about "**the**" landmark

There are often many…

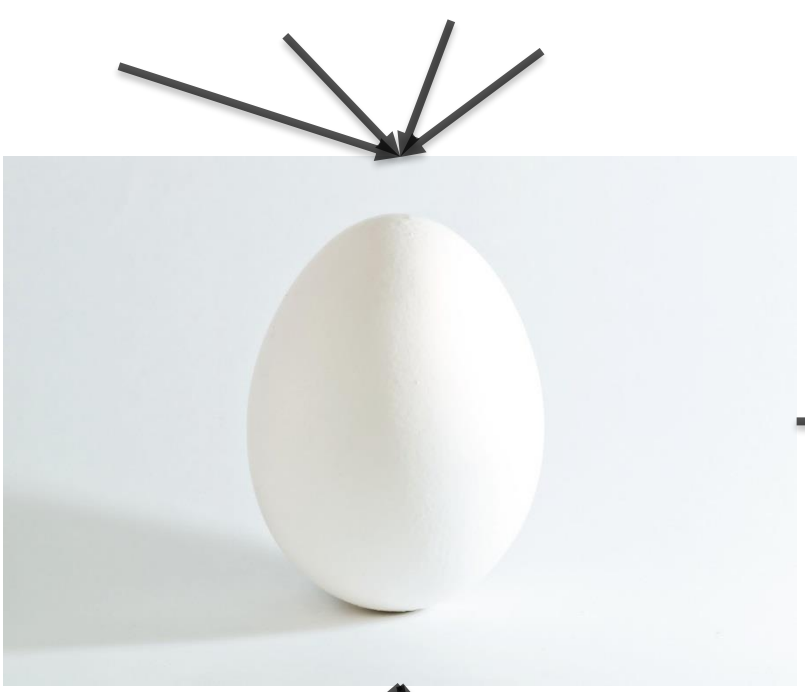


A landmark is **not** a unique location
(unlike real-world landmarks):
The intuition breaks down…

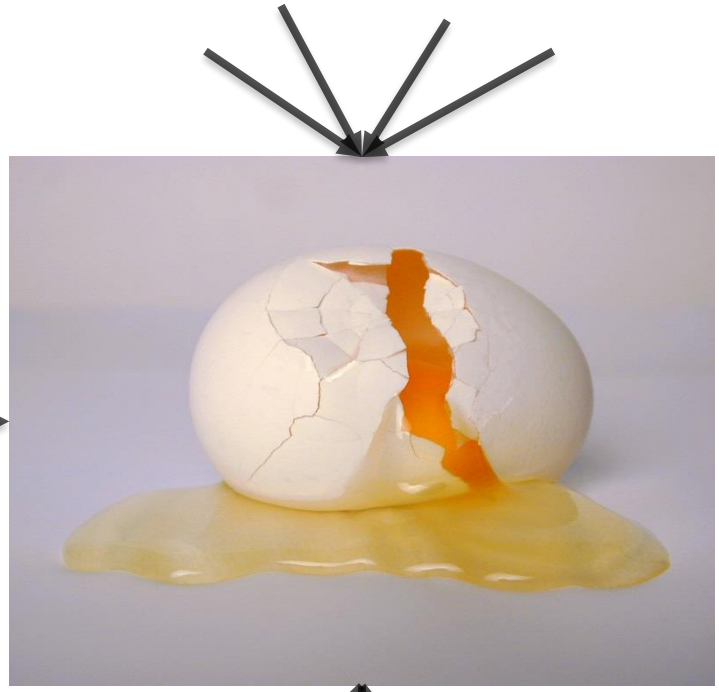
Landmarks: Ideas that Don't Work

jonkv@ida

”If I reach a state satisfying a landmark, I won’t have to backtrack”



crack(egg5)



All goal states are <u>here</u>
(uncracked egg)

Other states satisfying
landmark have(milk)