# AI Planning

## 6. Progression and Regression

## Should We Go Forward or Backward?

Álvaro Torralba, Cosmina Croitoru

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Winter Term 2018/2019

Thanks to Prof. Jörg Hoffmann for slide sources

# Agenda

1. Introduction
2. Progression
3. Regression
4. Pro and Contra
5. Conclusion

# What is "Search"?

**Here, we mean classical search:** (We'll usually omit the "classical".)

- A search space specifies a start search state, a target-identification function, and a successor search-state function.
- Find a path of search-state transitions from the start state to a state identified as a target.

→ Search state ≠ world state! E.g. regression, cf. later.

→ Classical search is the same as in AI'18 Chapters 4 and 5. But, there, we didn't worry about the search space and just assumed forward search.

# Planning as Classical Search: Choices

**There are three independent choices to make:**

## Choice 1: Search Space

- Progression.
  → Search forward from initial state to goal.
  → Search states = world states.
- Regression.
  → Search backward from goal to initial state.
  → Search states = sub-goals we would need to achieve.

→ **This Chapter**

# Planning as Classical Search: Choices, ctd.

**There are three independent choices to make:**

## Choice 2: Search Algorithm

- Blind search.
  $\rightarrow$ Depth-first, breadth-first, iterative depth-first, ...
- Heuristic search (systematic). Aka informed search (systematic).
  $\rightarrow$ A$^*$, IDA$^*$, ...
- Heuristic search (local). Aka informed search (local).
  $\rightarrow$ Hill-climbing, simulated annealing, beam search, ...

$\rightarrow$ **Next Chapter**

# Planning as Classical Search: Choices, ctd.

**There are three independent choices to make:**

## Choice 3: Search Control

- Heuristic function. (For heuristic searches.)
  → Critical-path heuristics, delete-relaxation heuristics, abstraction heuristics, landmarks heuristics, ...
- Pruning techniques.
  → Helpful actions pruning, symmetry elimination, dominance pruning, partial-order reduction.

→ **Chapters 8–20**

# Planning as Classical Search: Example Satisficing Systems

## FF [Hoffmann and Nebel (2001)] → Chapter 9

1. Search space: Progression.
2. Search algorithm: Enforced hill-climbing (informed local).
3. Search control: Delete-relaxation heuristic $h^{\mathsf{FF}}$ helpful actions pruning (HA) (incomplete).

## LAMA [Richter and Westphal (2010)] → Chapters 9, 14

1. Search space: Progression.
2. Search algorithm: GBFS with multiple queues (informed systematic).
3. Search control: $h^{\mathsf{FF}}$, landmarks, HA (complete).

## Mercury [Katz and Hoffmann (2014)] → Chapters 10, 14

1. Search space: Progression.
2. Search algorithm: GBFS with multiple queues (informed systematic).
3. Search control: Red-black heuristic, landmarks, HA (complete).

# Planning as Classical Search: Example Optimal Systems

**Fast Downward + abstraction heuristics** → **Chapters 11-13**

1. Search space: Progression.
2. Search algorithm: $\mathrm{A}^*$ (informed systematic).
3. Search control: Abstraction heuristics (admissible).

**Fast Downward + LMcut [Helmert and Domshlak (2009)]** → **Chapter 17**

1. Search space: Progression.
2. Search algorithm: $\mathrm{A}^*$ (informed systematic).
3. Search control: Abstraction heuristics (admissible).

**Symbolic Bidirectional Search [Kissmann and Edelkamp (2011); Torralba *et al.* (2017)]**

1. Search space: Progression and Regression.
2. Search algorithm: (Symbolic) Bidirectional Uniform-cost search.
3. Search control: $h^2$ mutexes (→ **Chapters 8**)

→ Fast Downward has become the main implementation basis for heuristic search planning. Its current version implements a great variety of techniques. Our **Programming Exercises** are based on it, too.

# What is a "Search Space"?

## Search Space for Classical Search

A (classical) search space is defined by the following three operations:

- start(): Generate the start (search) state.
- is-target($s$): Test whether a given search state is a target state.
- succ($s$): Generates the successor states $(a, s')$ of search state $s$, along with the actions through which they are reached.

$\rightarrow$ Search state $\neq$ world state! E.g. regression, cf. later.

$\rightarrow$ Progression and regression instantiate this template in different ways.

# Our Agenda for This Chapter

2. **Progression:** The (very) simple definition.
3. **Regression:** The less simple definition, for STRIPS vs. FDR and the differences between these two.
4. **Pro and Contra:** So which one should we use?

# Progression

**. . . is another word for Forward Search:**

## Search Space for Progression

Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The progression search space of $\Pi$ is given by:

- start$() = I$
- is-target$(s) = \begin{cases} \textbf{true} & \text{if } G \subseteq s \\ \textbf{false} & \text{otherwise} \end{cases}$
- succ$(s) = \{(a, s') \mid \Theta_\Pi \text{ has the transition } s \xrightarrow{a} s'\}$

The same definition applies to FDR tasks $\Pi = (V, A, c, I, G)$.

$\rightarrow$ Start from initial state, and apply actions until a goal state is reached.

$\rightarrow$ Search space = state space $\Rightarrow$ called state space search.

# Regression

**. . . is another word for Backward Search:**

## Search Space for Regression

Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The regression search space of $\Pi$ is given by:

- start$() = G$
- is-target$(g) = \begin{cases} \textbf{true} & \text{if } g \subseteq I \\ \textbf{false} & \text{otherwise} \end{cases}$
- succ$(g) = \{(a, g') \mid g' = regr(g, a)\}$

The *same definition* applies to FDR tasks $\Pi = (V, A, c, I, G)$.

$\rightarrow$ Start at goal, and regress over actions to produce subgoals, until a subgoal is contained in the initial state.

**Condition (*) required:** If $g' = regr(g, a)$, then for all $s'$ with $s' \models g'$, we have $s'[\![a]\!] = s$ where $s \models g$.

# Regressing Subgoals Over Actions: FDR

→ Intuition: $a$ can make the conjunctive subgoal $g$ true if (i) it achieves part of $g$; (ii) it contradicts none of $g$; and (iii) the new subgoal we would have to solve is not self-contradictory.

**Definition (FDR Regression).** *Let $(V, A, c, I, G)$ be an FDR planning task, $g$ be a partial variable assignment, and $a \in A$.*
*We say that $g$ is regressable over $a$ if*

(i) $\mathit{eff}_a \cap g \neq \emptyset$;

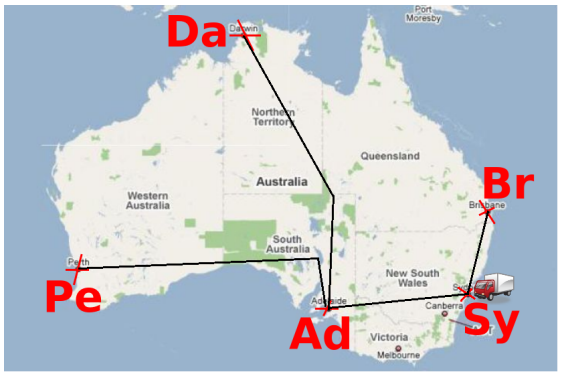
(ii) *there is no $v \in V$ s.t. $v \in V[\mathit{eff}_a] \cap V[g]$ and $\mathit{eff}_a(v) \neq g(v)$; and*

(iii) *there is no $v \in V$ s.t. $v \notin V[\mathit{eff}_a]$, $v \in V[\mathit{pre}_a] \cap V[g]$, and $\mathit{pre}_a(v) \neq g(v)$.*

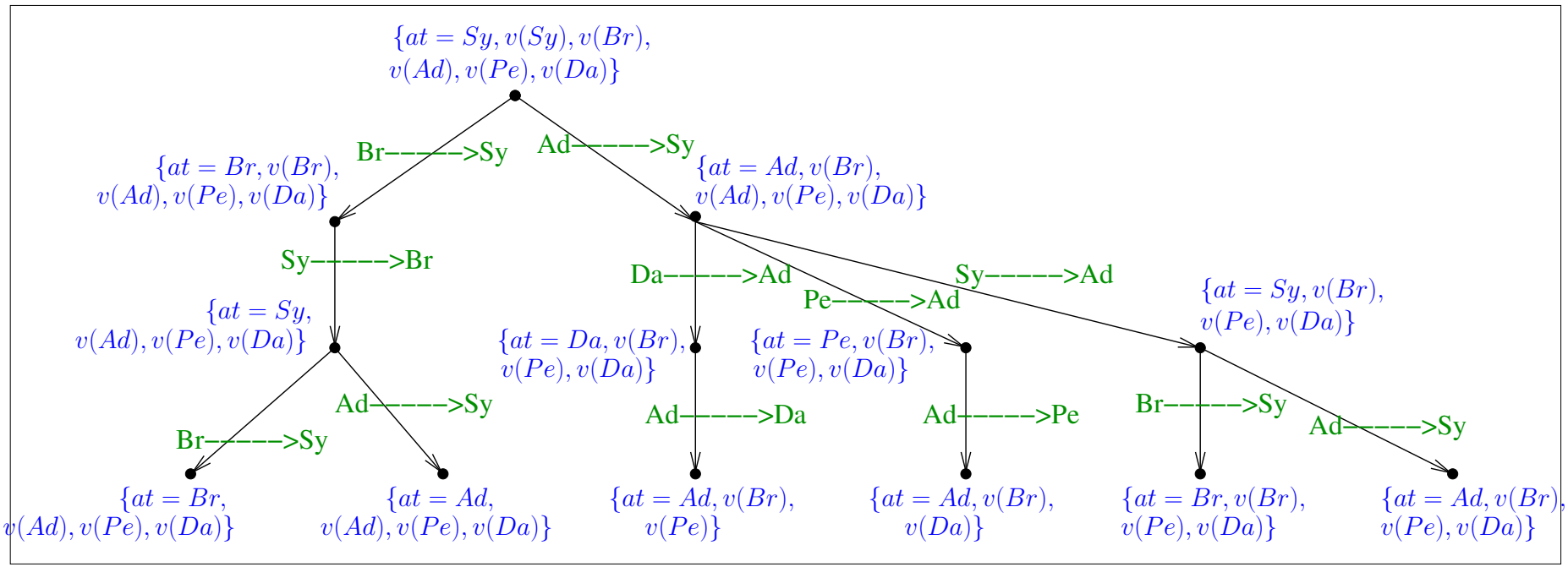
*In that case, the regression of $g$ over $a$ is $\mathit{regr}(g, a) = (g \setminus \mathit{eff}_a) \cup \mathit{pre}_a$; else $\mathit{regr}(g, a)$ is undefined, written $\mathit{regr}(g, a) = \bot$.*

**Proposition.** *This definition of $\mathit{regr}$ satisfies condition (*) on slide 15.*

# Regression Example: "TSP" in FDR




- Variables $V$: $at : \{Sy, Ad, Br, Pe, Da\}$; $v(x) : \{T, F\}$ for $x \in \{Sy, Ad, Br, Pe, Da\}$.
- Actions $a \in A$: $drive(x, y)$ where $x, y$ have a road.
- Initial state $I$: $at = Sy, v(Sy) = T, v(x) = F$ for $x \neq Sy$.
- Goal $G$: $at = Sy, v(x) = T$ for all $x$.



# Regressing Subgoals Over Actions: STRIPS

**Definition (STRIPS Regression).** *Let $(P, A, c, I, G)$ be a STRIPS planning task, $g \subseteq P$, and $a \in A$. We say that $g$ is regressable over $a$ if*

(i) $add_a \cap g \neq \emptyset$; *and*

(ii) $del_a \cap g = \emptyset$.

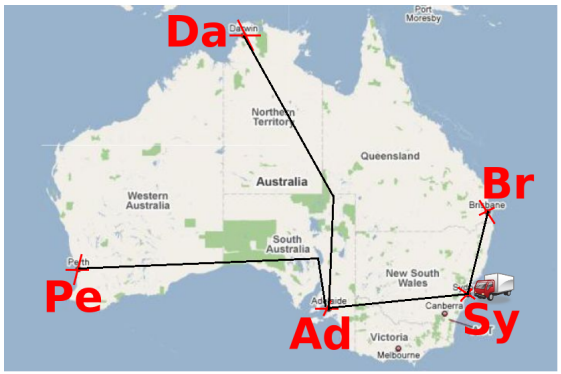
*In that case, the regression of $g$ over $a$ is $regr(g, a) = (g \setminus add_a) \cup pre_a$; else $regr(g, a)$ is undefined, written $regr(g, a) = \bot$..*

**Proposition.** *This definition of $regr$ satisfies condition (*) on slide 15.*

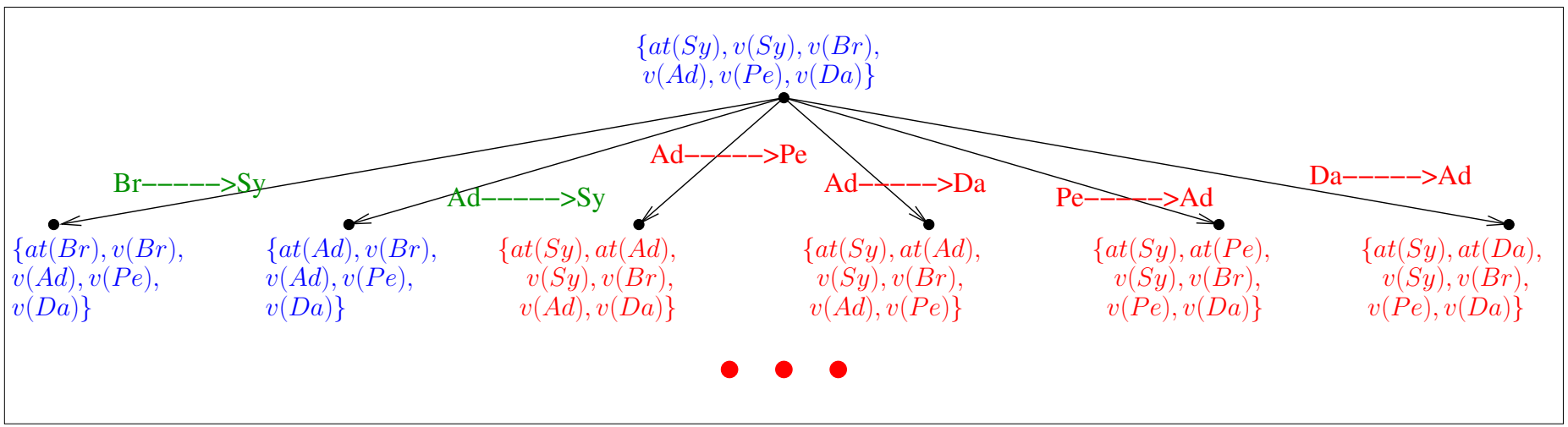
**Note the difference to FDR:**

- In (ii), instead of "contradicting variable values" we only look at the action's immediate deletes.
  $\rightarrow$ This is weaker because we fail to see, e.g., that different truck positions yield contradictions as well (see next slide).
- (iii) here is missing completely because in STRIPS there is no possibility for a subgoal to be "self-contradictory".
  $\rightarrow$ This is weaker because we fail to see, e.g., that subgoals requiring several different truck positions are self-contradictory (see next slide).

# Regression Example: "TSP" in STRIPS

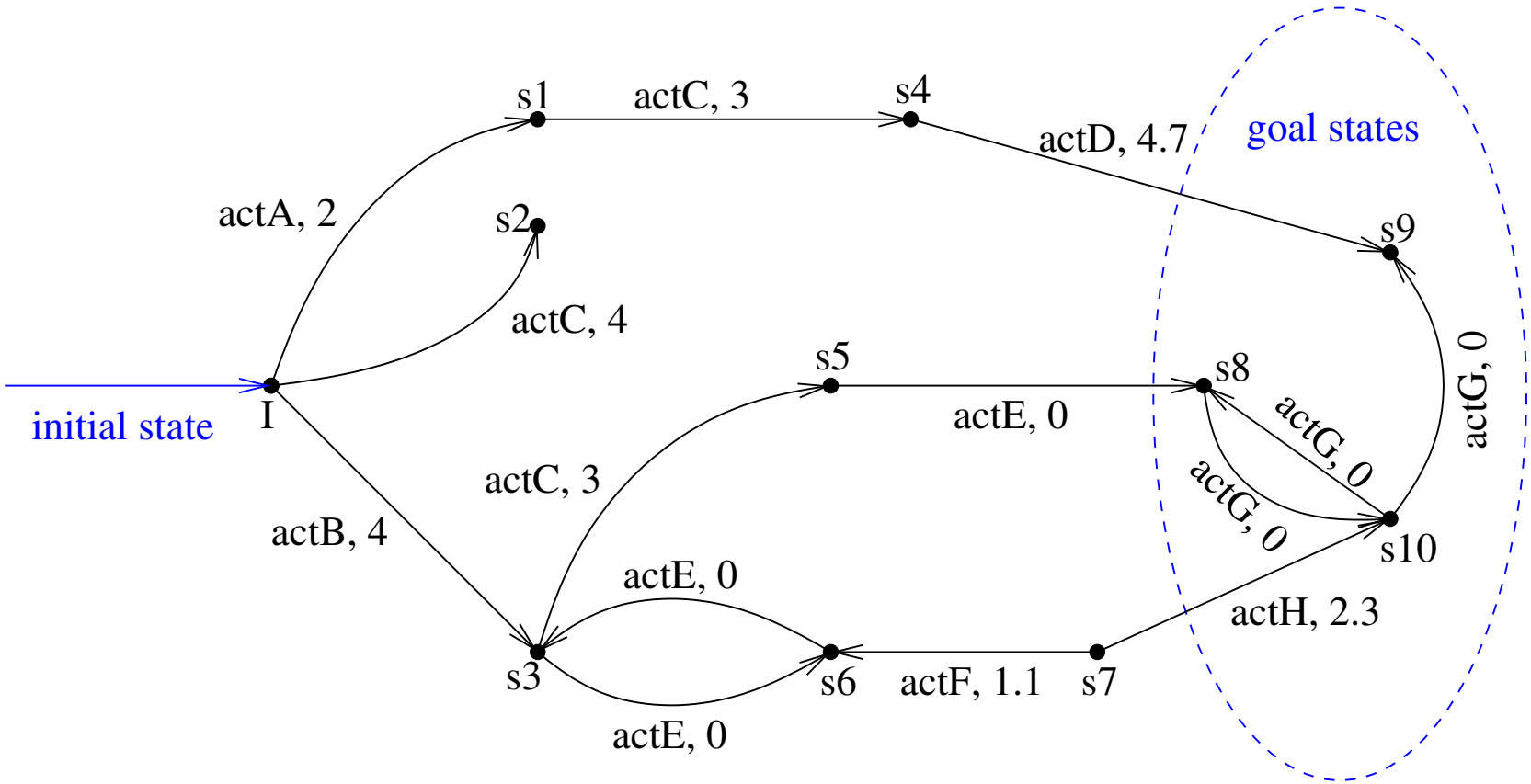



- Propositions $P$: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- Actions $a \in A$: $drive(x, y)$ where $x, y$ have a road.
- Initial state $I$: $at(Sy), v(Sy)$.
- Goal $G$: $at(Sy), v(x)$ for all $x$.




→ Reminder **Chapter 2**: "In regression, FDR allows to avoid myriads of unreachable search states."

# Reachable and Solvable States in Progression

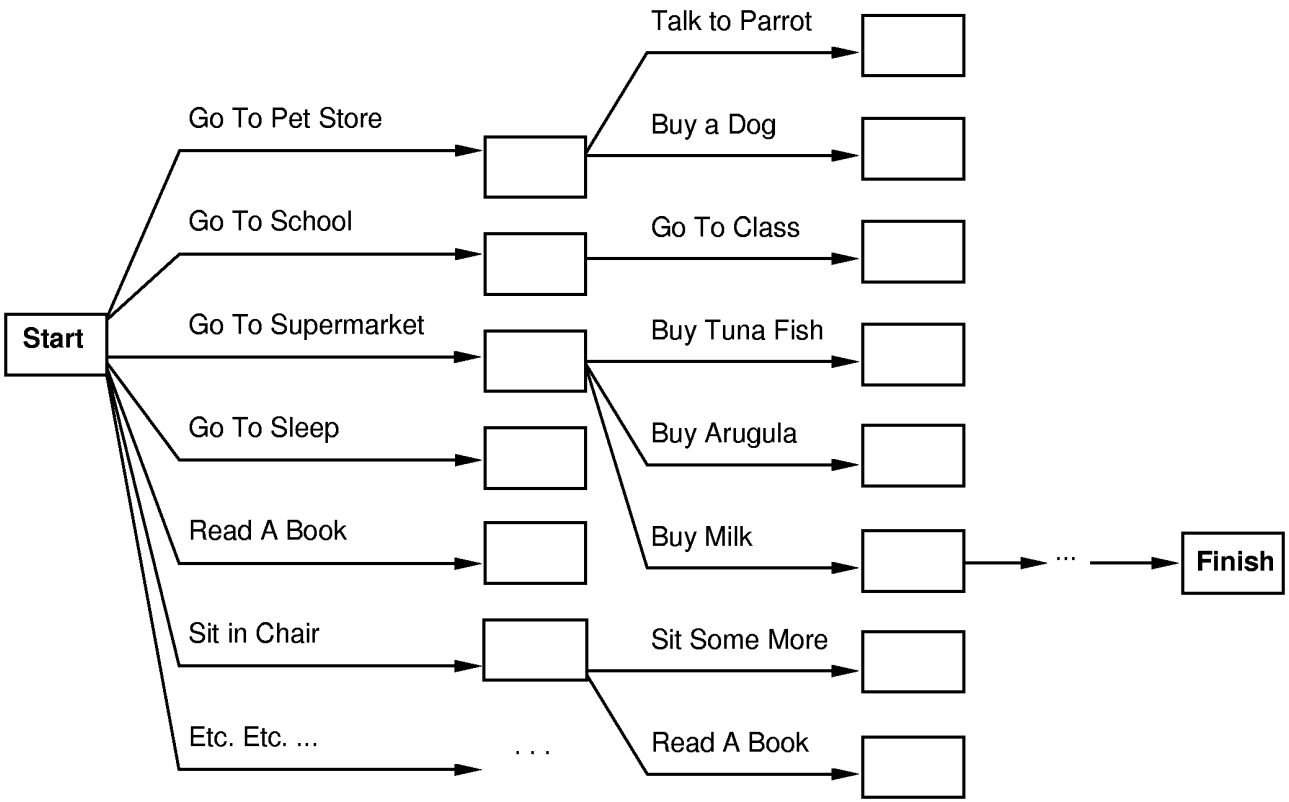



- Does progression explore the state "at bottom, 2nd from right"? No.
- Does progression explore the state "near top, 2nd from left"? Yes.

→ Progression explores only reachable states, but may explore unsolvable ones.

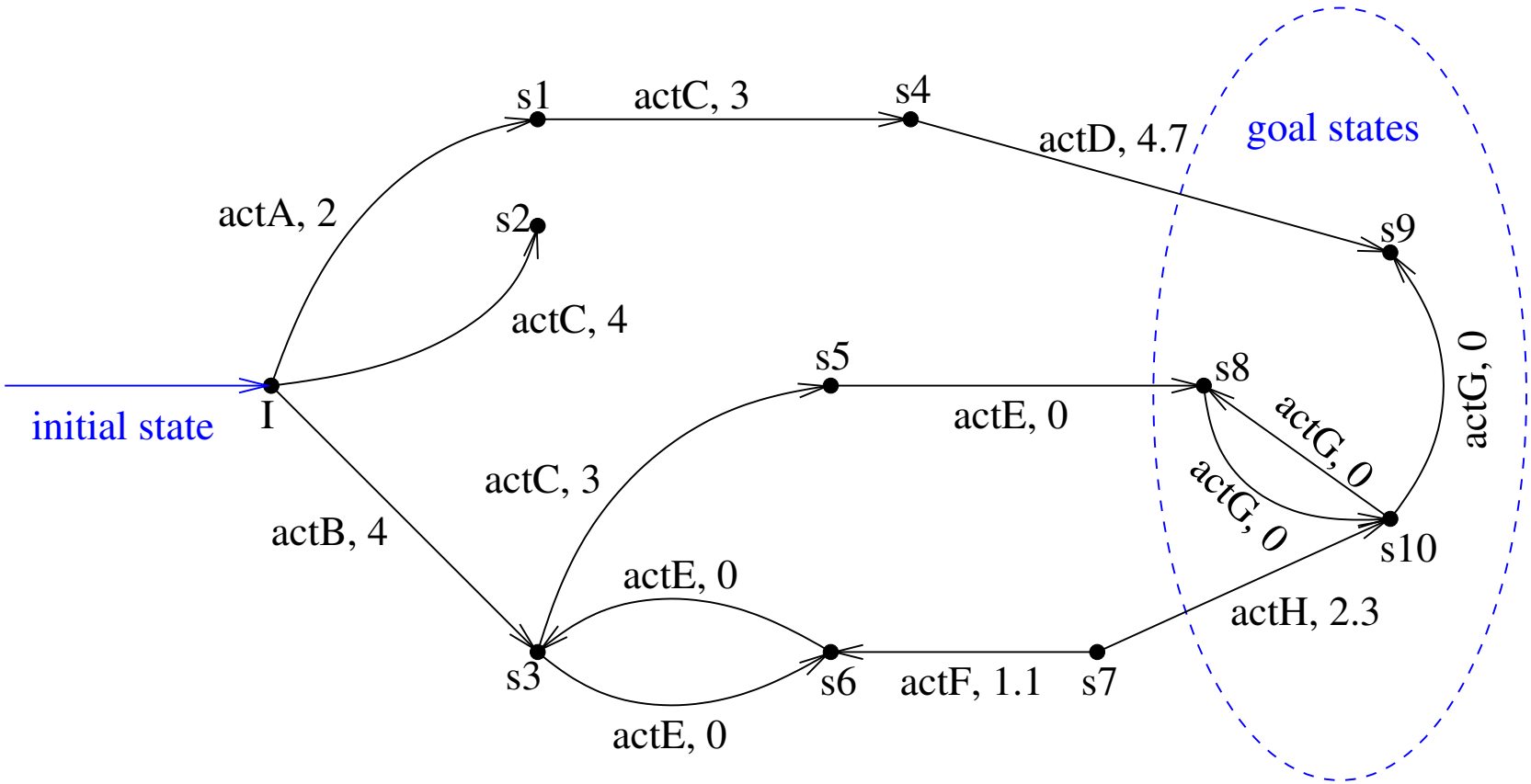
# Progression and Relevance

**Observe:** Progression doesn't know what's "relevant", i.e., what contributes to reaching the goal:




→ Use heuristic function to guide the search towards the goal!

# Reachable and Solvable States in Regression




- Does regression explore the state "near top, 2nd from left"? No.
- Does regression explore the state "at bottom, 2nd from right"? Yes.

→ Regression explores only solvable states, but may explore unreachable ones.

# Regression and Reachability

**Observe:** Regression doesn't know what's "reachable", i.e., what contributes to reaching the initial state:


→ Use heuristic function to guide the search towards the initial state!

# Questionnaire

→ Regarding fruitless parts of the forward search space:

**Question!**

**In which of these STRIPS domains can the state space contain unsolvable (dead-end) but reachable states?**

(A): "Logistics" (B): "TSP" in Australia

(C): 15-Puzzle (D): FreeCell

→ (A): No, because the transition relation is invertible. From any reachable state, we can go back to the initial state and take it from there.

→ (B): No. The transitions are not invertible, but this does no harm (nothing can go fatally wrong).

→ (C): Same as (A). (There are *unreachable* dead ends, though: The state space of the 15-Puzzle falls into two disconnected parts.)

→ (D): Yes, we might choose fatally wrong moves.

# Questionnaire

→ Regarding fruitless parts of the backward search space: (recall that the "state space" contains all subsets of facts)

**Question!**

**In which of these STRIPS domains can the state space contain solvable (non-dead-end) but unreachable states?**

(A): "TSP" in Australia
(B): "Logistics"
(C): 15-Puzzle
(D): FreeCell

→ (A): Yes. For example, the state $s = \{at(Sy), at(Ad), v(Sy), v(Br), v(Ad), v(Da)\}$ is solvable because applying $drive(Ad, Pe)$ to $s$ yields a goal state.

→ (B): Yes. For example, the state $s = \{truck(B), truck(D), pack1(D)\}$ is solvable because applying $drive(B, A)$ to $s$ yields a goal state.

→ (C), (D): Similar, we can always include some additional true facts that don't prevent the goal from being reached but that do prevent $s$ from being reachable.

# So Which One Should We Use?

**In favor of progression:**

- Regression has in the past often had serious trouble getting lost in gazillions of solvable but unreachable states.

  Reachable dead end states tend to be less frequent in practice (cf. the two previous slides).
- Progression allows easy formulation of searches for more complex planning formalisms (numbers, durations, uncertainty, you name it).
- Basically all current heuristic search planners, including Fast Downward which you will use in the **Exercises**, use progression.

→ We assume progression for the rest of the course.

**That said:**

- Which one works better depends on the input task and search algorithm, and there is no comprehensive understanding of this.
- Regression is a building block in **Chapters 8, 9, 10, 14, and 16**.

# Summary

- Search is required in planning because the problem is computationally hard.
- We consider classical search, that finds a path through a search space implicitly defined in terms of the operations start(), is-target($s$), and succ($s$).
- Progression is forward search from the initial state to the goal, in the state space. To be effective, it needs to be informed about relevance.
- Regression is backward search from the goal to the initial state, in a space of subgoals that correspond to sets of world states. To be effective, it needs to be informed about reachability.
- FDR regression is a lot more effective than STRIPS regression, because its search space contains less unreachable states.

# Remarks

- Regression used to be the dominating approach to planning, until about the year 2000.
- The argument was basically the one on slide 22. Why nobody saw the obvious counter-argument of slide 24? I have no idea.

- STRIPS regression is equivalent to FDR regression when pruning unreachable subgoals by invariants.
- That is why Graphplan [Blum and Furst (1995)] beat everybody in 1995! (cf. AI'18 Chapter 14 "The History of Planning")

  $\rightarrow$ To get this straight: People were taking domains with finite-domain variables (truck position), and modeling them in STRIPS (one fact for each truck location) where the invariance information (exactly one of these facts is true in any reachable world state) is not explicit. Consequently, their regression planners died exploring myriads of subgoals requiring the same variable to have more than one possible value. Graphplan fixed this by automatically recovering the invariance information from STRIPS input. Obviously, it would be much easier to simply use FDR input in the first place.

# References I

Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. In S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1636–1642, Montreal, Canada, August 1995. Morgan Kaufmann.

Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In Alfonso Gerevini, Adele Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 162–169. AAAI Press, 2009.

Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

Michael Katz and Jörg Hoffmann. Mercury planner: Pushing the limits of partial delete relaxation. In *IPC 2014 planner abstracts*, pages 43–47, 2014.

# References II

Peter Kissmann and Stefan Edelkamp. Improving cost-optimal domain-independent symbolic planning. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the 25th National Conference of the American Association for Artificial Intelligence (AAAI'11)*, pages 992–997, San Francisco, CA, USA, July 2011. AAAI Press.

Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39:127–177, 2010.

Álvaro Torralba, Vidal Alcázar, Peter Kissmann, and Stefan Edelkamp. Efficient symbolic search for cost-optimal planning. *Artificial Intelligence*, 242:52–79, 2017.