Automated Planning

EMATM0042 – Intelligent Information Systems

Thursday 14 March

kevin.mcareavey@bristol.ac.uk

Previous Lecture…

1. Agents
    - Agent function
    - Agent program

2. Task environments
    - Problem Analysis
    - Properties

3. Types of agents
    - Reactive agents
    - Proactive agents
    - Cognitive agents

This Lecture…

1. Classical planning
    - Planning languages

2. Planning under uncertainty
    - Conformant planning
    - Contingent planning
    - Markov decision processes

3. Online planning
    - Classical replanning

Automated Planning & Agents

- Why is automated planning important to agent design?
  - Generating an agent function?
  - Generating plans for goal- or utility-based agents?
  - Generating plans for BDI agents?

- Certain or uncertain initial state?
- (Non-)deterministic or stochastic transitions?
- Full, partial, or no observability?
- Goals or utilities?

- Technique?
- Online or offline?
- Optimal or satisficing?
- Approximate?
- Anytime?

- Sequence?
- Partially-ordered set?
- Tree?
- Graph?
- Function?

Classical Planning
Definition

- Classical planning problem $(\mathcal{S}, \mathcal{A}, s_1, T, G)$
  - Initial state $s_1 \in \mathcal{S}$
  - Deterministic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$
  - Set of goal states $G \subseteq \mathcal{S}$

- A plan (or solution) is a finite sequence of actions (from $\mathcal{A}$)
  - Typically interested in some notion of an optimal plan
  - A single optimal plan is sufficient

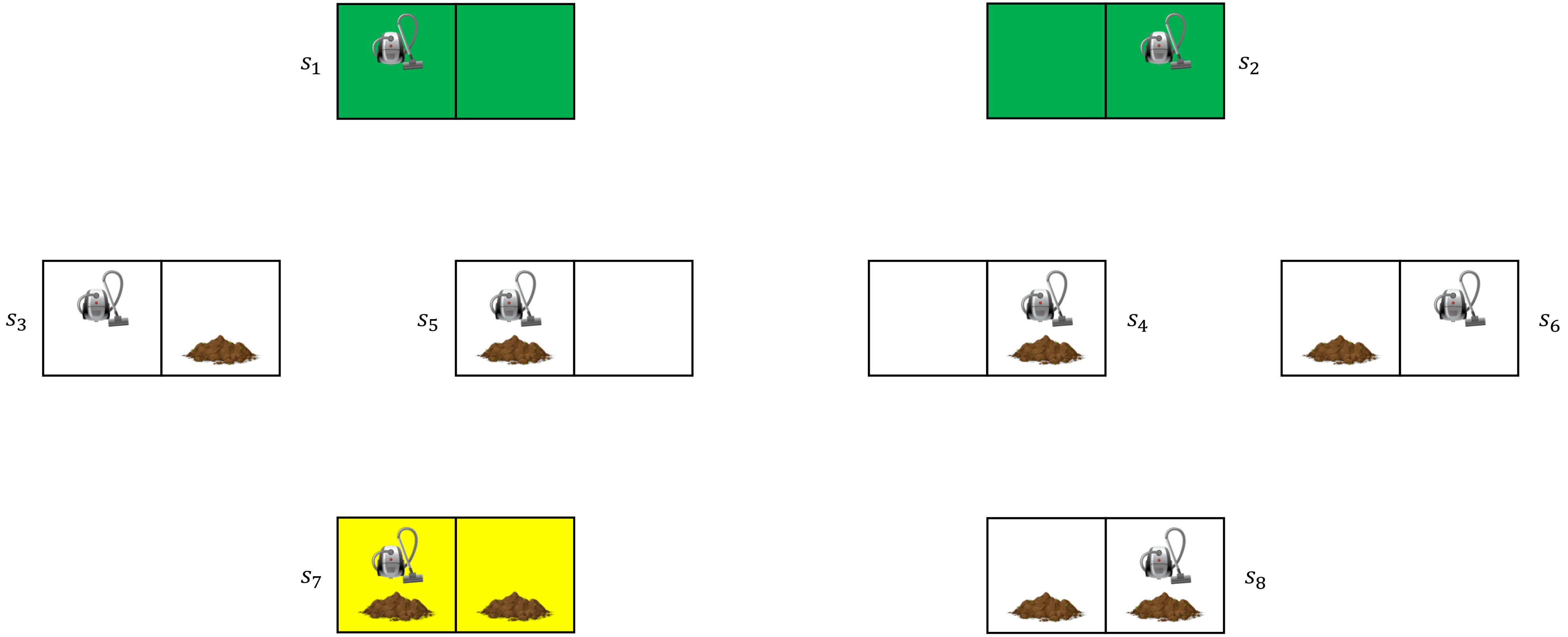
Classical Planning

Example (1)

Set of states $\mathcal{S} = \{s_1, s_2, \dots, s_8\}$

Initial state s_7

Goal states $G = \{s_1, s_2\}$

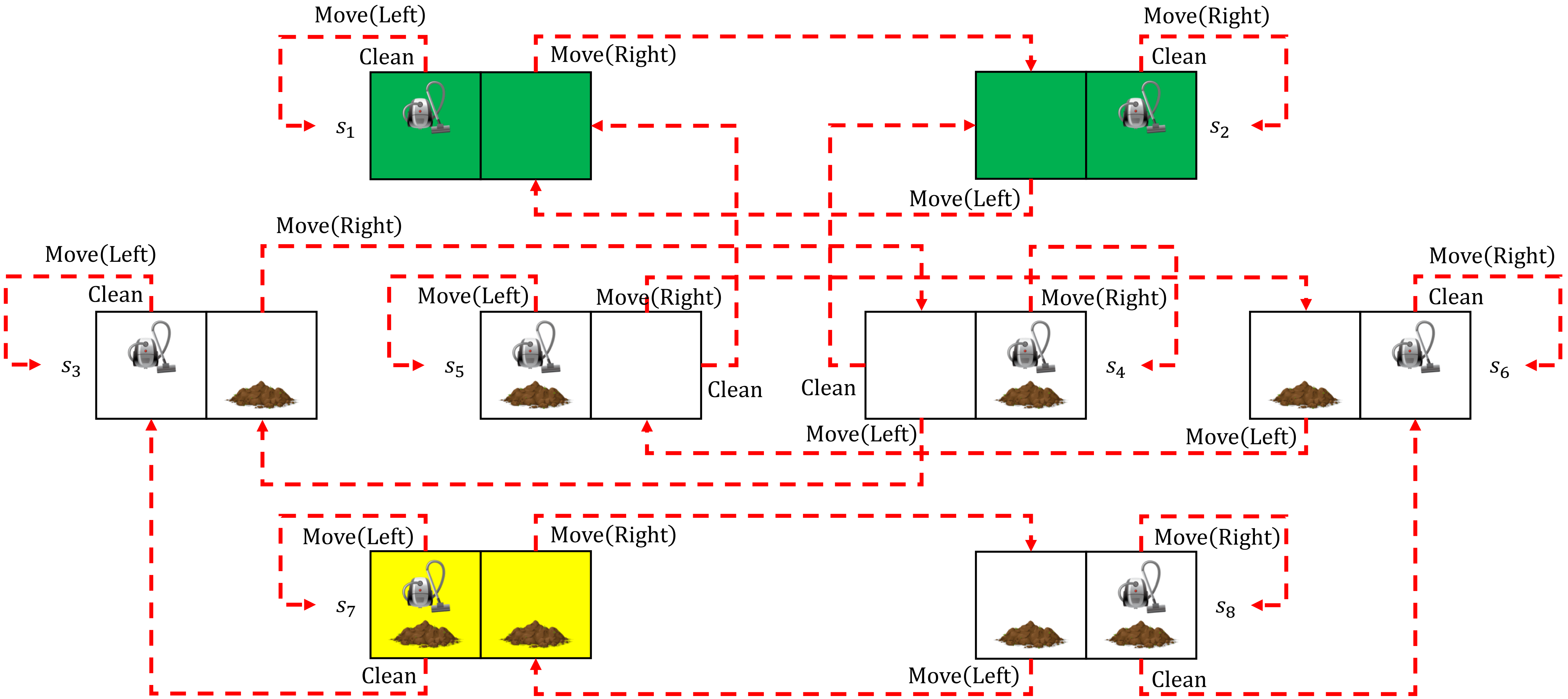


Classical Planning

Example (2)

Set of actions $\mathcal{A} = \{\text{Move(Left)}, \text{Move(Right)}, \text{Clean}\}$

Transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$

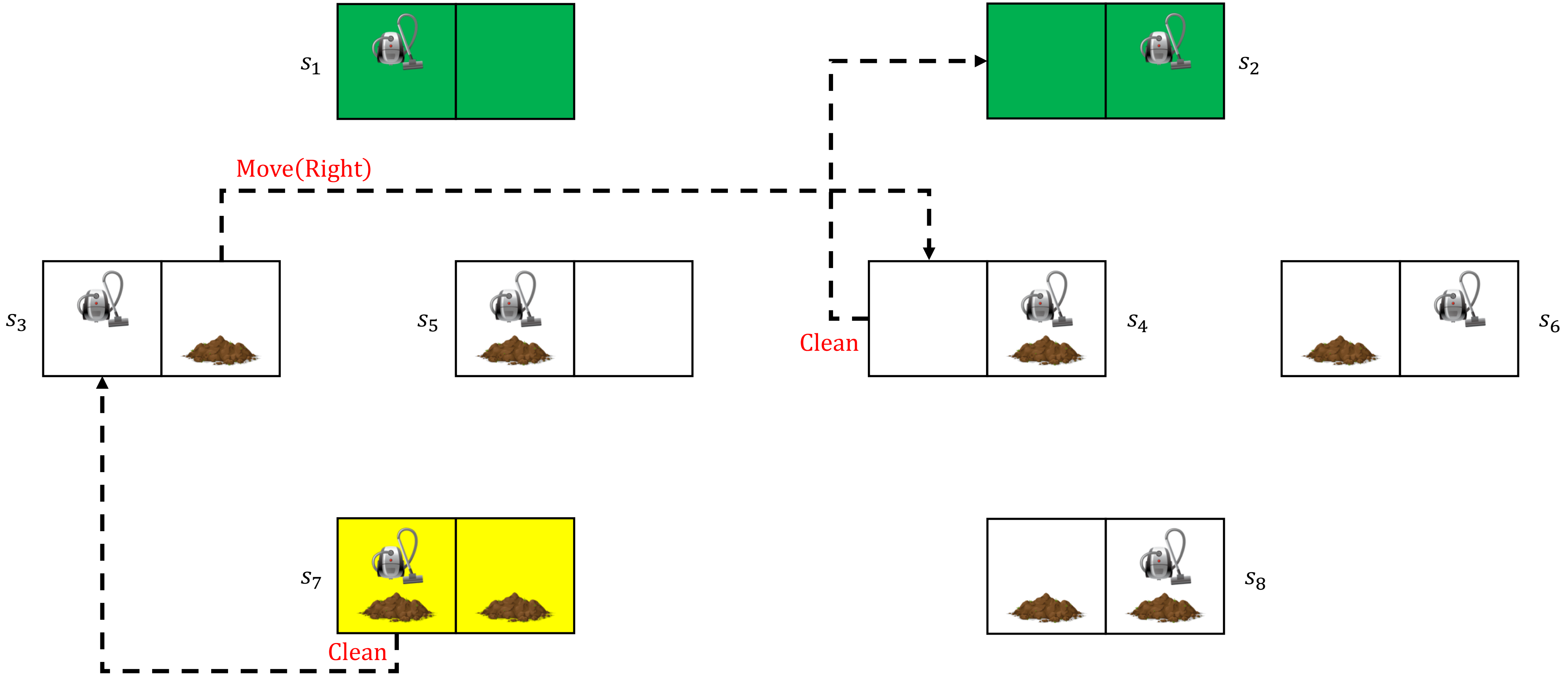


Classical Planning

Example (3)

Set of actions $\mathcal{A} = \{\text{Move(Left)}, \text{Move(Right)}, \text{Clean}\}$

Plan $\pi_1 = (\text{Clean}, \text{Move(Right)}, \text{Clean})$



Classical Planning

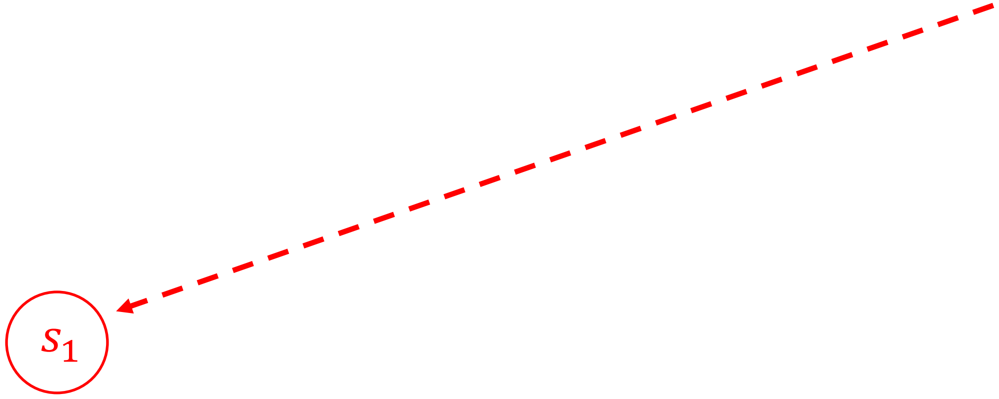
Solution Form (1)

1. Finite sequence of actions $p = (a_1, a_2, \ldots, a_n)$ from initial state $s_1 \in \mathcal{S}$

Classical Planning

Solution Form (2)

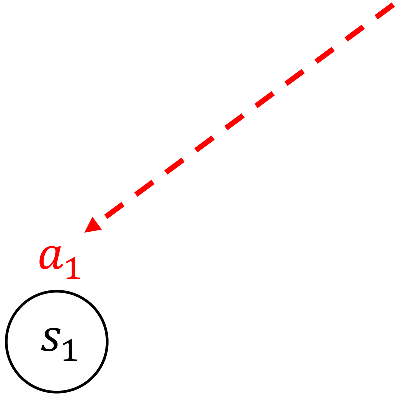
1. Finite sequence of actions $p = (a_1, a_2, \dots, a_n)$ from initial state $s_1 \in \mathcal{S}$



Classical Planning

Solution Form (3)

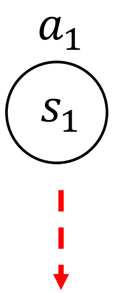
1. Finite sequence of actions $p = (a_1, a_2, \dots, a_n)$ from initial state $s_1 \in \mathcal{S}$



Classical Planning

Solution Form (4)

1. Finite sequence of actions $p = (a_1, a_2, \dots, a_n)$ from initial state $s_1 \in \mathcal{S}$

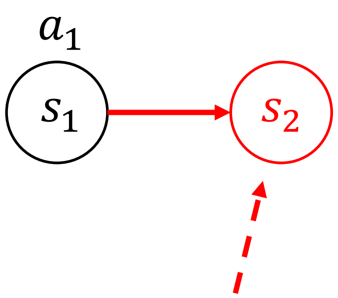


Successor state $T(s_1, a_1) = s_2$

Classical Planning

Solution Form (5)

1. Finite sequence of actions $p = (a_1, a_2, \dots, a_n)$ from initial state $s_1 \in \mathcal{S}$

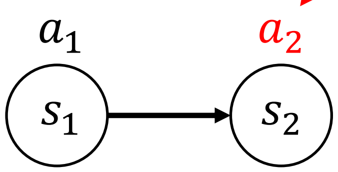


Successor state $T(s_1, a_1) = s_2$

Classical Planning

Solution Form (6)

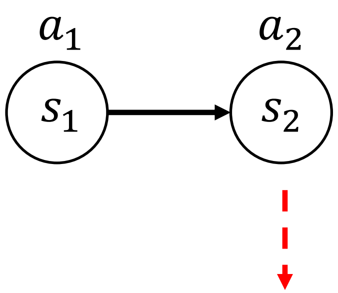
1. Finite sequence of actions $p = (a_1, a_2, \ldots, a_n)$ from initial state $s_1 \in \mathcal{S}$



Classical Planning

Solution Form (7)

1. Finite sequence of actions $p = (a_1, a_2, \dots, a_n)$ from initial state $s_1 \in \mathcal{S}$

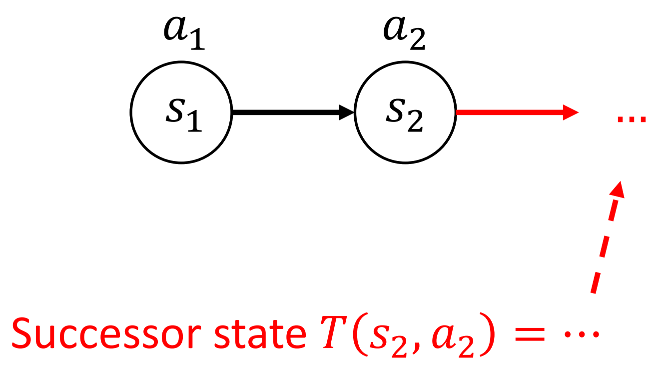


Successor state $T(s_2, a_2) = \cdots$

Classical Planning

Solution Form (8)

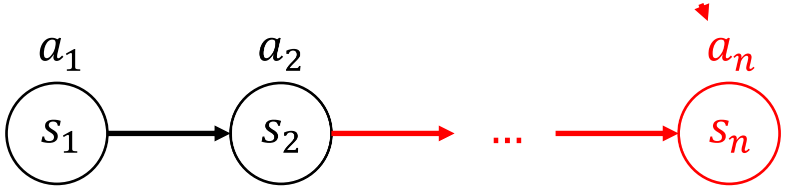
1. Finite sequence of actions $p = (a_1, a_2, \dots, a_n)$ from initial state $s_1 \in \mathcal{S}$



Classical Planning

Solution Form (9)

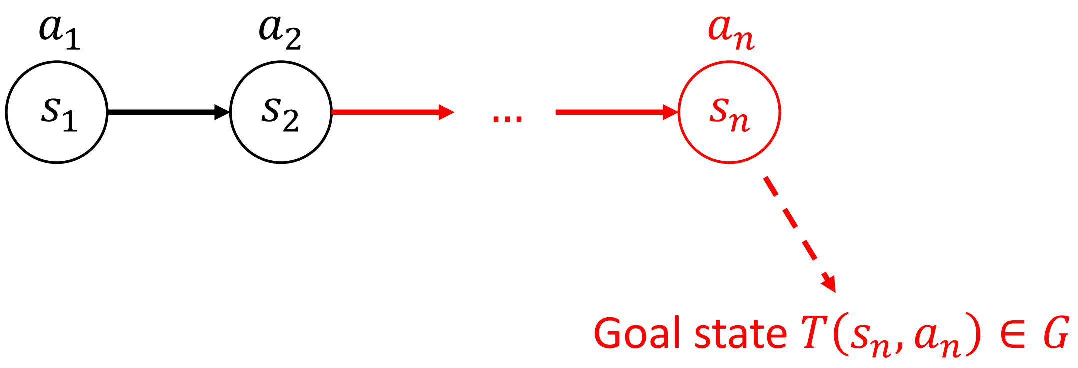
1. Finite sequence of actions $p = (a_1, a_2, \dots, a_n)$ from initial state $s_1 \in \mathcal{S}$



Classical Planning

Solution Form (10)

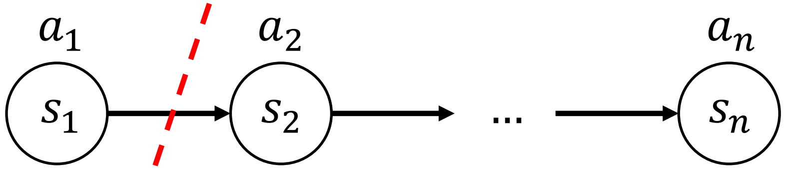
1. Finite sequence of actions $p = (a_1, a_2, \ldots, a_n)$ from initial state $s_1 \in \mathcal{S}$



Classical Planning

Solution Form (11)

1. Finite sequence of actions $p = (a_1, a_2, \dots, a_n)$ from initial state $s_1 \in \mathcal{S}$

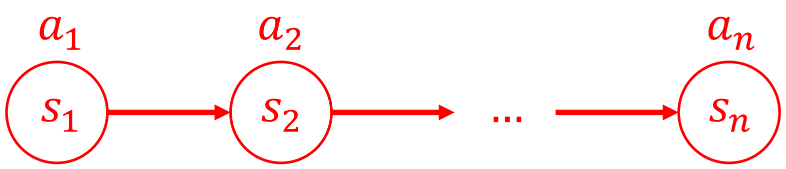


2. Partial state-based policy $\pi_{\mathcal{S}} : \mathcal{S}' \to \mathcal{A}$ with $\mathcal{S}' \subseteq \mathcal{S}$ such that $\pi_{\mathcal{S}}$ is closed with respect to $s_1 \in \mathcal{S}$
    - If $s \in \mathcal{S}$ is reachable from s_1 via $\pi_{\mathcal{S}}$ and T, then $\pi_{\mathcal{S}}$ is closed with respect s_1 if and only if $s \in \mathcal{S}'$

Classical Planning

Solution Form (12)

1. Finite sequence of actions $p = (a_1, a_2, \dots, a_n)$ from initial state $s_1 \in \mathcal{S}$



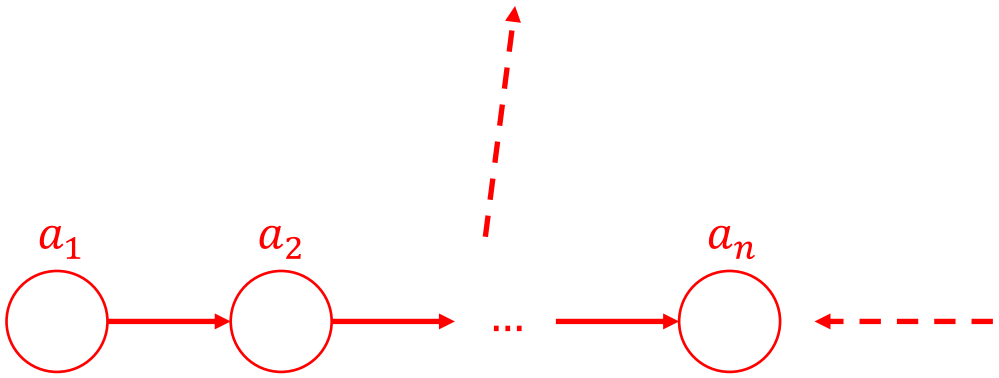
3. Policy graph $(\mathcal{S}', E, s_1, \pi_{\mathcal{S}})$
- Directed edges $E \subseteq \mathcal{S}' \times \mathcal{S}'$
- Rooted at s_1
- Nodes labelled by $\pi_{\mathcal{S}}$

2. Partial state-based policy $\pi_{\mathcal{S}} : \mathcal{S}' \rightarrow \mathcal{A}$ with $\mathcal{S}' \subseteq \mathcal{S}$ such that $\pi_{\mathcal{S}}$ is closed with respect to $s_1 \in \mathcal{S}$
- If $s \in \mathcal{S}$ is reachable from s_1 via $\pi_{\mathcal{S}}$ and T, then $\pi_{\mathcal{S}}$ is closed with respect s_1 if and only if $s \in \mathcal{S}'$

Classical Planning

Solution Form (13)

1. Finite sequence of actions $p = (a_1, a_2, \ldots, a_n)$ from initial state $s_1 \in \mathcal{S}$



3. Policy graph $(\mathcal{S}', E, s_1, \pi_{\mathcal{S}})$
   - Directed edges $E \subseteq \mathcal{S}' \times \mathcal{S}'$
   - Rooted at s_1
   - Nodes labelled by $\pi_{\mathcal{S}}$

2. Partial state-based policy $\pi_{\mathcal{S}} : \mathcal{S}' \rightarrow \mathcal{A}$ with $\mathcal{S}' \subseteq \mathcal{S}$ such that $\pi_{\mathcal{S}}$ is closed with respect to $s_1 \in \mathcal{S}$
   - If $s \in \mathcal{S}$ is reachable from s_1 via $\pi_{\mathcal{S}}$ and T, then $\pi_{\mathcal{S}}$ is closed with respect s_1 if and only if $s \in \mathcal{S}'$

Classical Planning

Planning Languages

- STRIPS
  - States and transition functions are defined implicitly via state variables and action descriptions
  - Historically important – basis for all(?) modern planning languages
- Action Description Language (ADL)
  - Influential set of extensions to STRIPS language
  - Notable extensions include the introduction of variables (as in FOL or logic programming) and conditional effects
- Planning Domain Definition Language (PDDL)
  - Current de facto standard planning language
  - Used by the International Planning Competition (IPC)
  - Many variants (e.g. 1.0, 3.1) and extensions for richer planning problems (e.g. NuPDDL, PPDDL, PO-PPDDL, MA-PDDL)

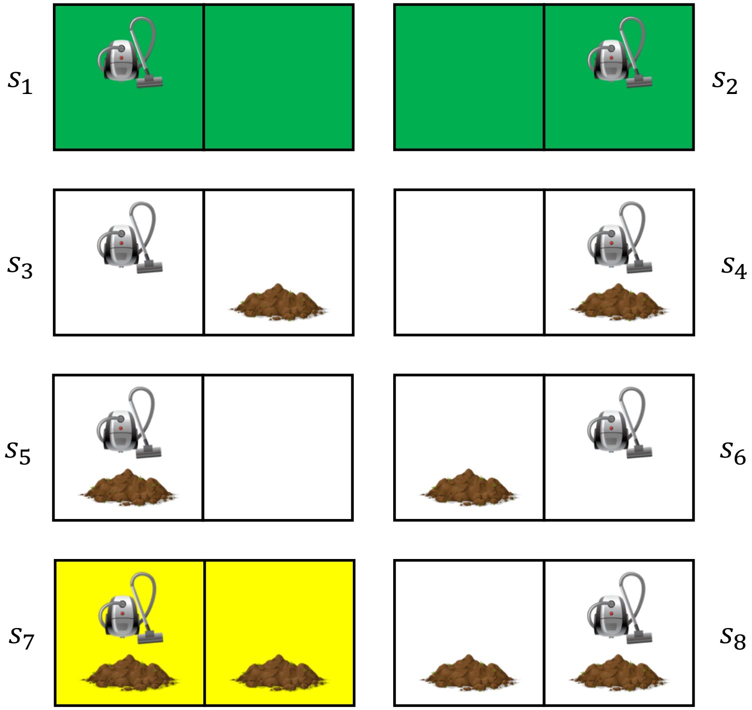
Classical Planning

STRIPS

- Set of propositional atoms $V = \{v, v', \dots\}$ such that $\mathcal{S} = 2^V$
  - $s \vDash v$ where $s \in \mathcal{S}$ if and only if $v \in s$
- Set of propositional formulas $\mathcal{L}$ over V with respect to $\{\neg, \wedge, \vee\}$
  - $\mathrm{mod}(\varphi) = \{s \in \mathcal{S} \mid s \vDash \varphi\}$ is the set of models of φ
  - Language may be restricted (e.g. to conjunctions of positive literals)
- Initial state $s_1 \in \mathcal{S}$
  - Set of atoms $s_1 \subseteq V$ that are true initially
- Set of action descriptions $D = \{(a, \mathrm{Pre}(a), \mathrm{Eff}^-(a), \mathrm{Eff}^+(a)) \mid a \in \mathcal{A}\}$
  - $\mathrm{Pre}(a) \in \mathcal{L}$ is the precondition for action $a \in \mathcal{A}$
  - $\mathcal{A}(s) = \{a \in \mathcal{A} \mid s \vDash \mathrm{Pre}(a)\}$ is the set of applicable actions in state $s \in \mathcal{S}$
  - $\mathrm{Eff}^-(a) \subseteq V$ is the "delete list" for action $a \in \mathcal{A}$
  - $\mathrm{Eff}^+(a) \subseteq V$ is the "add list" for action $a \in \mathcal{A}$
  - $T(s, a) = [s \setminus \mathrm{Eff}^-(a)] \cup \mathrm{Eff}^+(a)$ is the successor state after executing $a \in \mathcal{A}(s)$ in state $s \in \mathcal{S}$
  - $\mathrm{Eff}^-(a) \cap \mathrm{Eff}^+(a) = \emptyset$ ensures the add/delete lists are consistent
- Goal $\varphi \in \mathcal{L}$
  - $\mathrm{mod}(\varphi) \subseteq \mathcal{S}$ is the set of goal states

Classical Planning

STRIPS – Example



s_i	Agent(Left)	Dirt(Left)	Dirt(Right)
s_1	True	False	False
s_2	False	False	False
s_3	True	False	True
s_4	False	False	True
s_5	True	True	False
s_6	False	True	False
s_7	True	True	True
s_8	False	True	True

- Initial state:
  - $\{\text{Agent(Left)}, \text{Dirt(Left)}, \text{Dirt(Right)}\}$
- Goal:
  - $\neg\text{Dirt(Left)} \wedge \neg\text{Dirt(Right)}$
- Action $a_1 = \text{Move(Left)}$:
  - $\text{Pre}(a_1) = \top$
  - $\text{Eff}^-(a_1) = \emptyset$
  - $\text{Eff}^+(a_1) = \{\text{Agent(Left)}\}$
- Action $a_2 = \text{Move(Right)}$:
  - $\text{Pre}(a_2) = \top$
  - $\text{Eff}^-(a_2) = \{\text{Agent(Left)}\}$
  - $\text{Eff}^+(a_2) = \emptyset$
- Action $a_3 = \text{Clean(Left)}$:
  - $\text{Pre}(a_3) = \text{Agent(Left)}$
  - $\text{Eff}^-(a_3) = \{\text{Dirt(Left)}\}$
  - $\text{Eff}^+(a_3) = \emptyset$
- Action $a_4 = \text{Clean(Right)}$:
  - $\text{Pre}(a_4) = \neg\text{Agent(Left)}$
  - $\text{Eff}^-(a_4) = \{\text{Dirt(Right)}\}$
  - $\text{Eff}^+(a_4) = \emptyset$

Classical Planning

Technique: A* State-Space Search (1)

$\mathcal{S} = \{s_1, s_2, \dots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

$G = \{s_{17}, s_{18}\}$

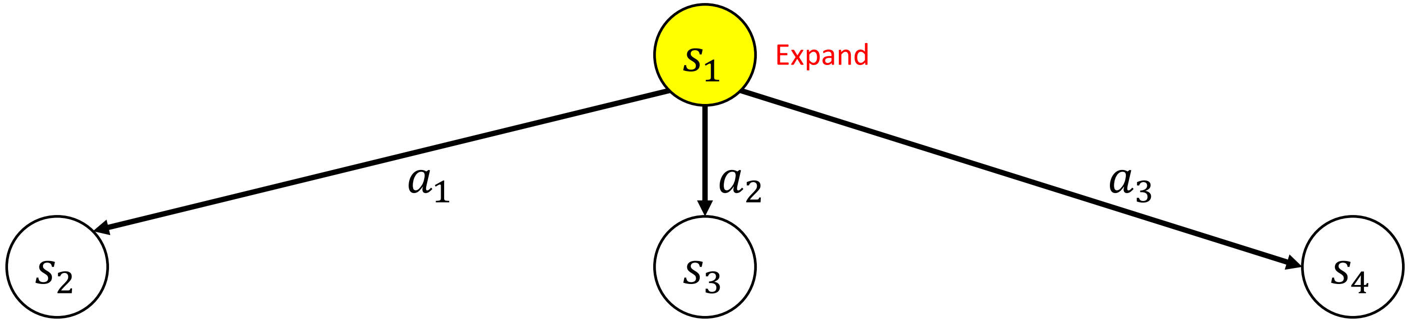
Classical Planning

Technique: A* State-Space Search (2)

$\mathcal{S} = \{s_1, s_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

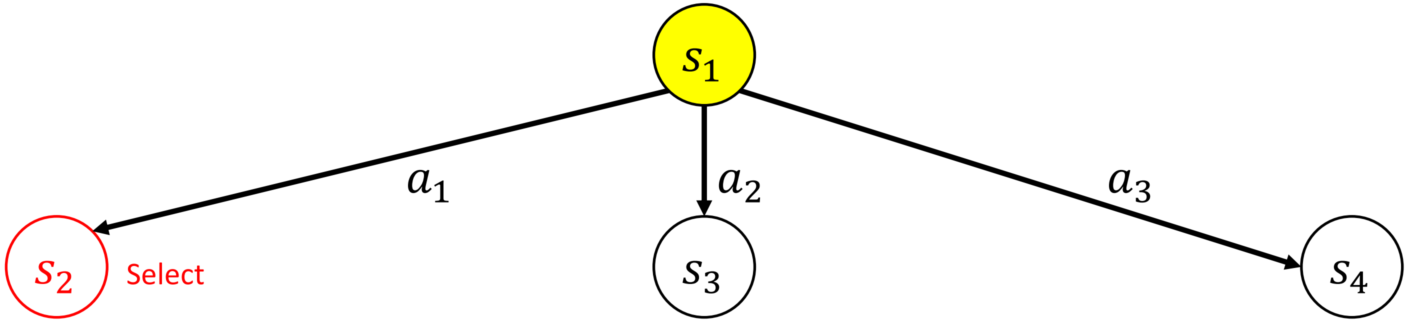
$G = \{s_{17}, s_{18}\}$



Classical Planning

Technique: A* State-Space Search (3)

$$\mathcal{S} = \{s_1, s_2, \ldots\}$$
$$\mathcal{A} = \{a_1, a_2, a_3\}$$
$$G = \{s_{17}, s_{18}\}$$



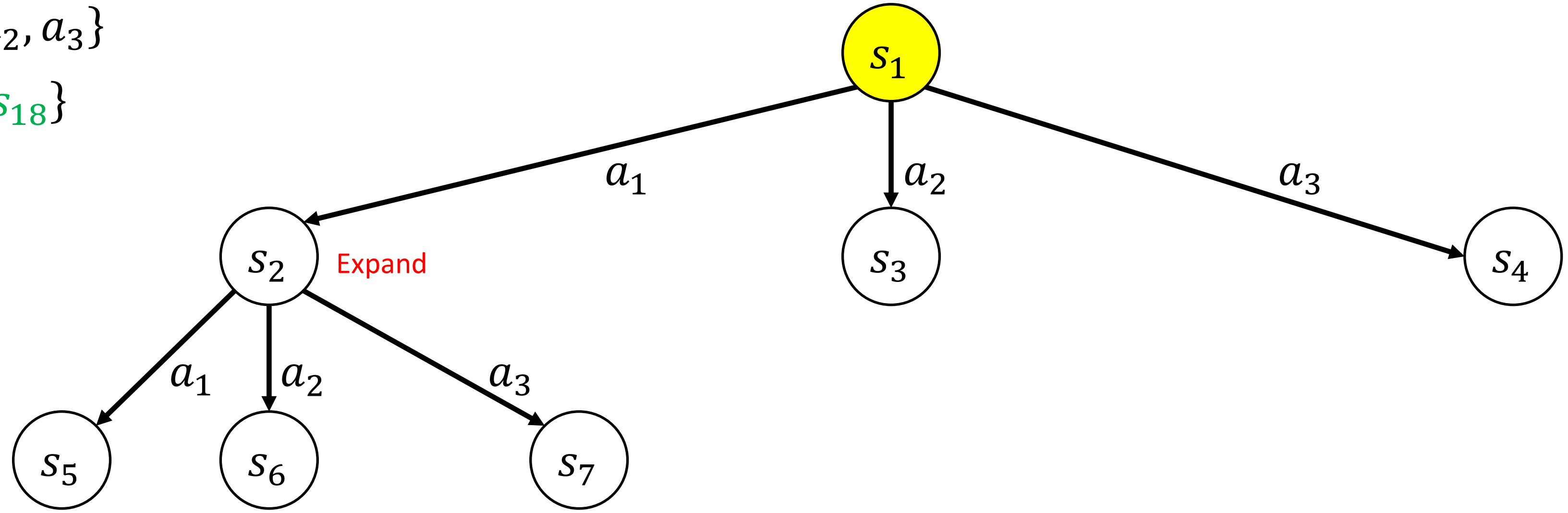
Classical Planning

Technique: A* State-Space Search (4)

$\mathcal{S} = \{s_1, s_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

$G = \{s_{17}, s_{18}\}$



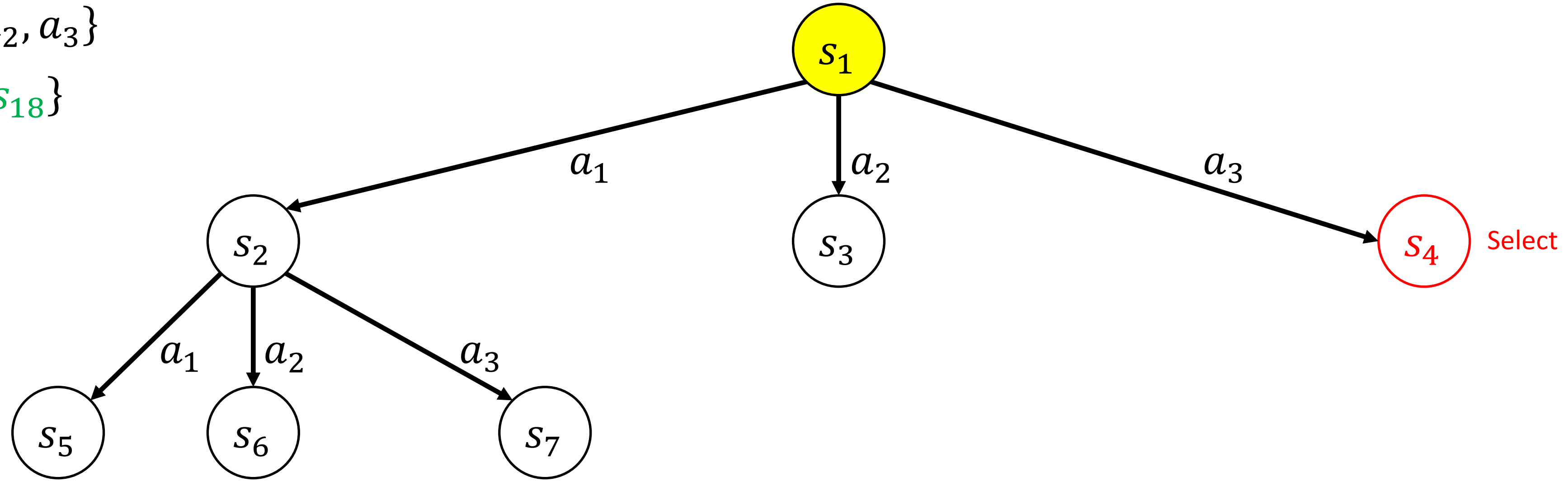
Classical Planning

Technique: A* State-Space Search (5)

$\mathcal{S} = \{s_1, s_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

$G = \{s_{17}, s_{18}\}$



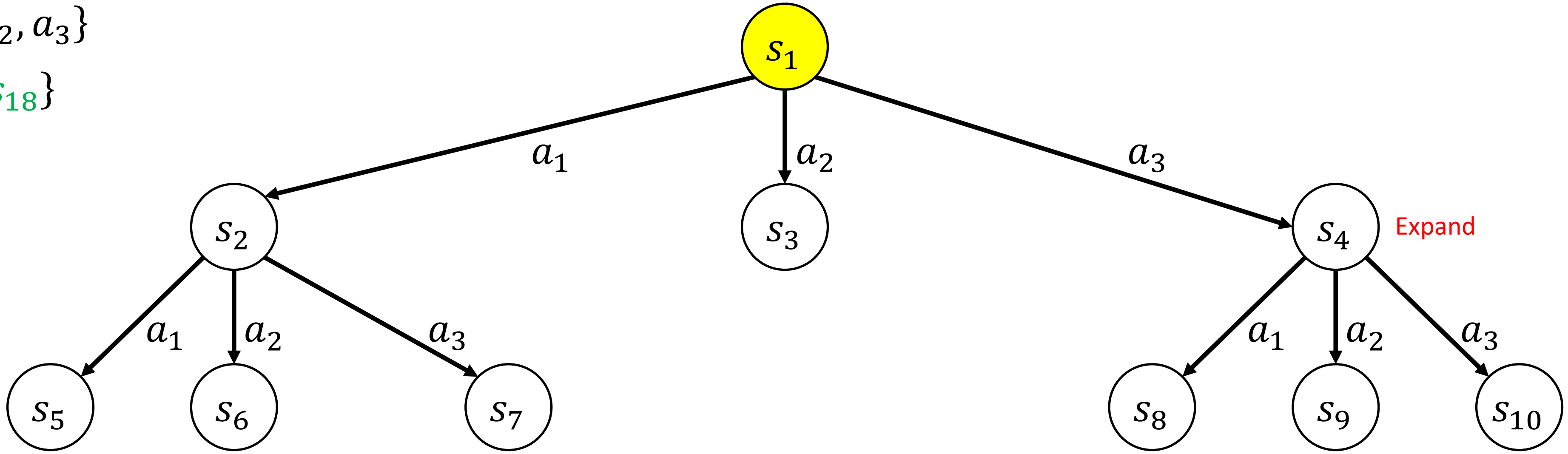
Classical Planning

Technique: A* State-Space Search (6)

$\mathcal{S} = \{s_1, s_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

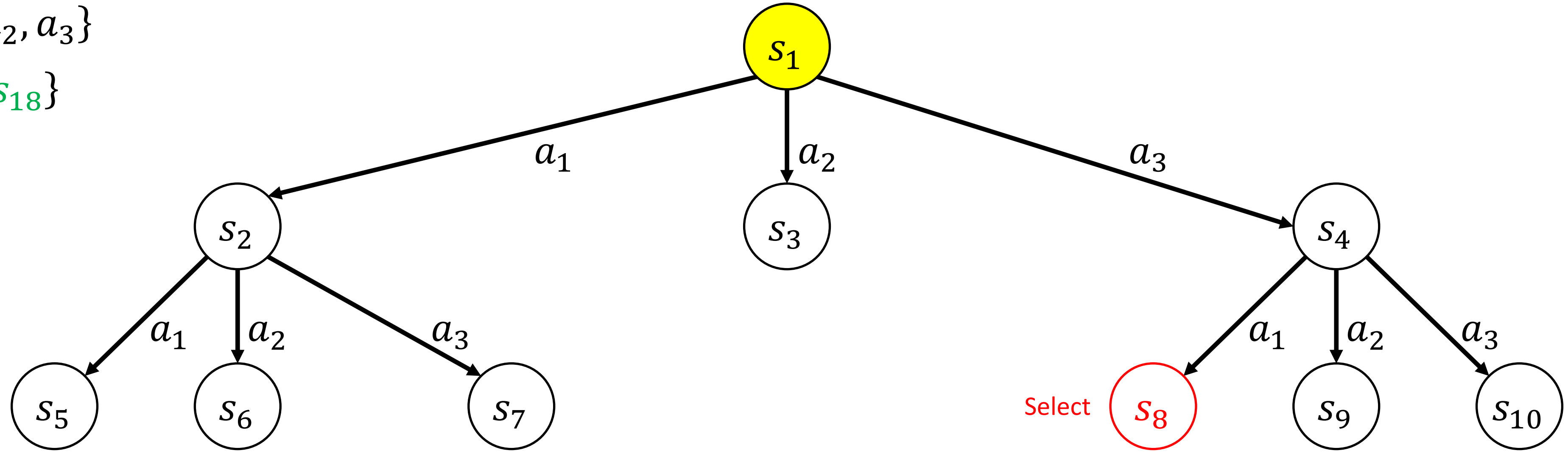
$G = \{s_{17}, s_{18}\}$



Classical Planning

Technique: A* State-Space Search (7)

$$\mathcal{S} = \{s_1, s_2, \ldots\}$$
$$\mathcal{A} = \{a_1, a_2, a_3\}$$
$$G = \{s_{17}, s_{18}\}$$



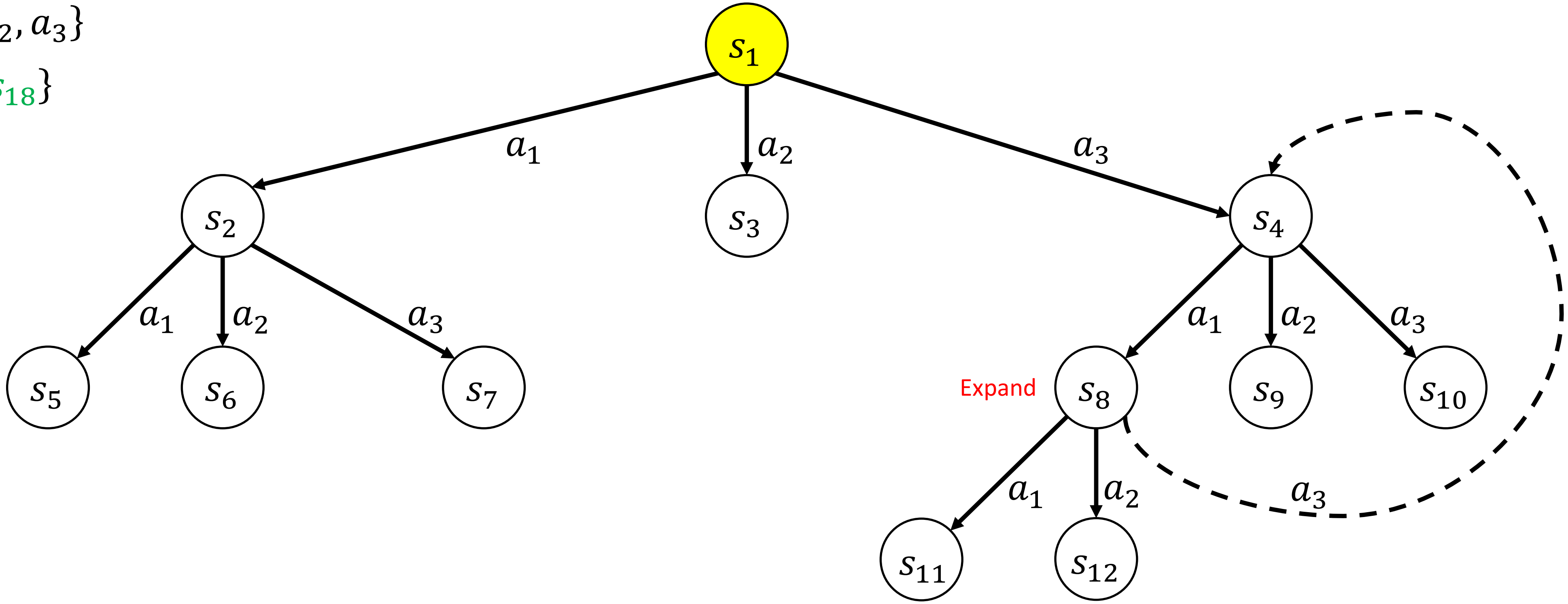
Classical Planning

Technique: A* State-Space Search (8)

$\mathcal{S} = \{s_1, s_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

$G = \{s_{17}, s_{18}\}$



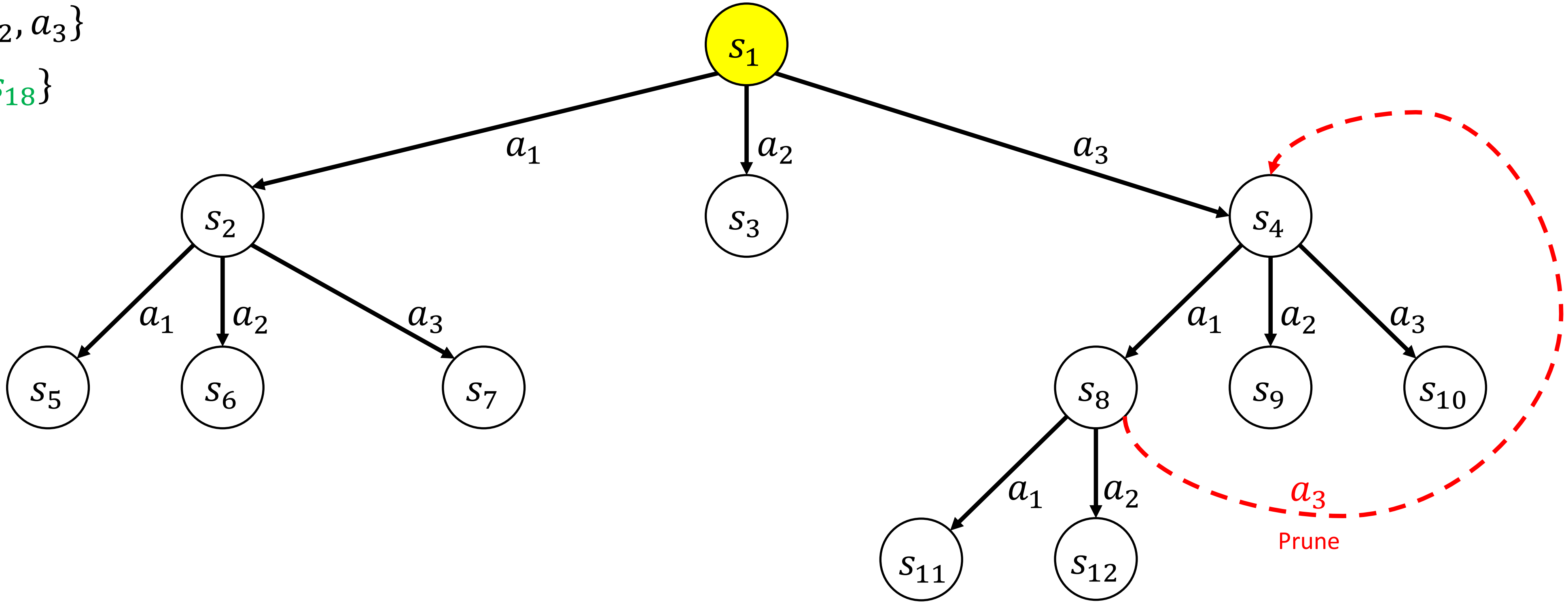
Classical Planning

Technique: A* State-Space Search (9)

$\mathcal{S} = \{s_1, s_2, \dots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

$G = \{s_{17}, s_{18}\}$



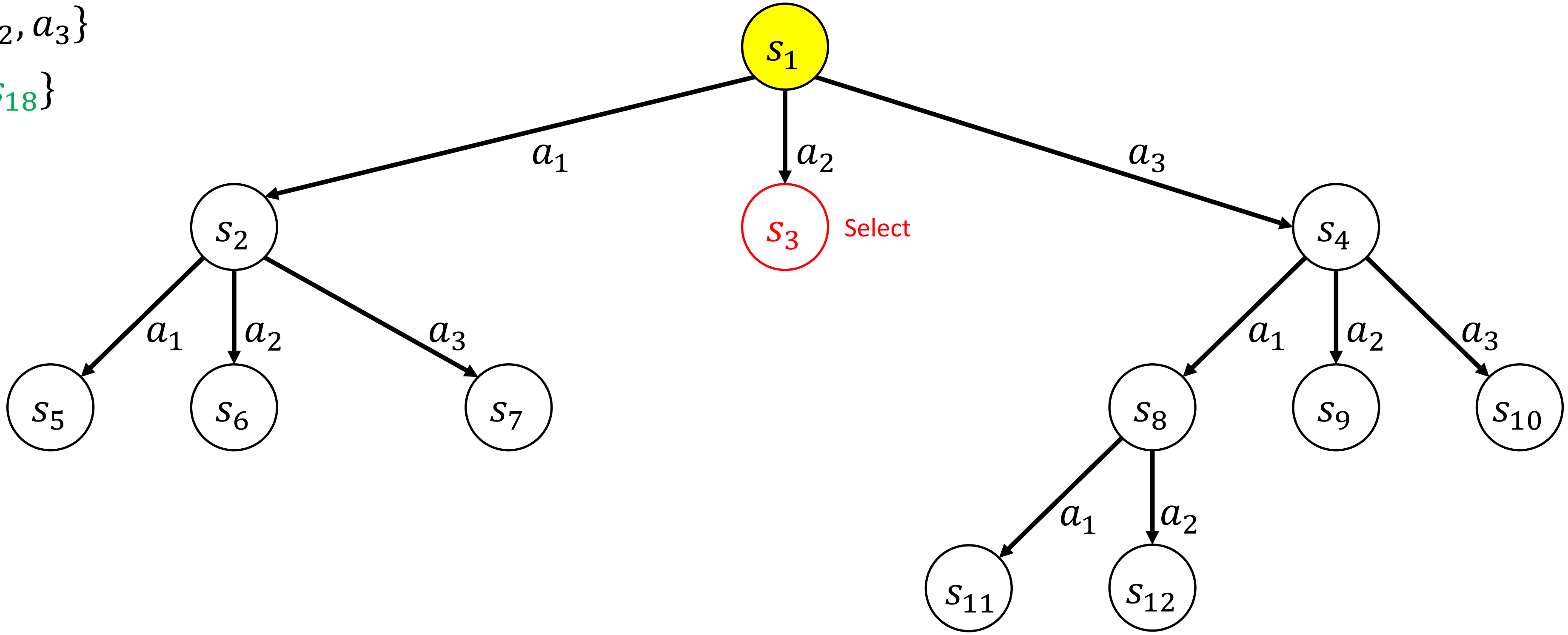
Classical Planning

Technique: A* State-Space Search (10)

$\mathcal{S} = \{s_1, s_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

$G = \{s_{17}, s_{18}\}$



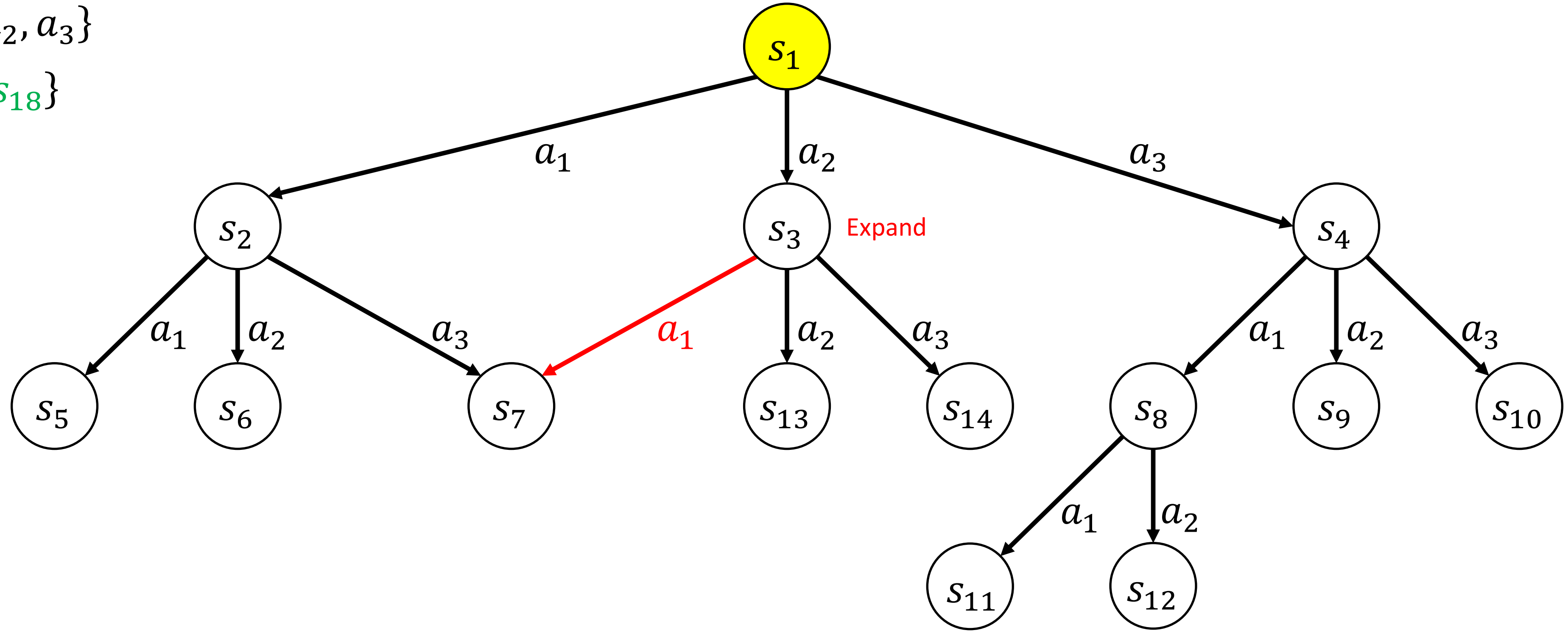
Classical Planning

Technique: A* State-Space Search (11)

$\mathcal{S} = \{s_1, s_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

$G = \{s_{17}, s_{18}\}$



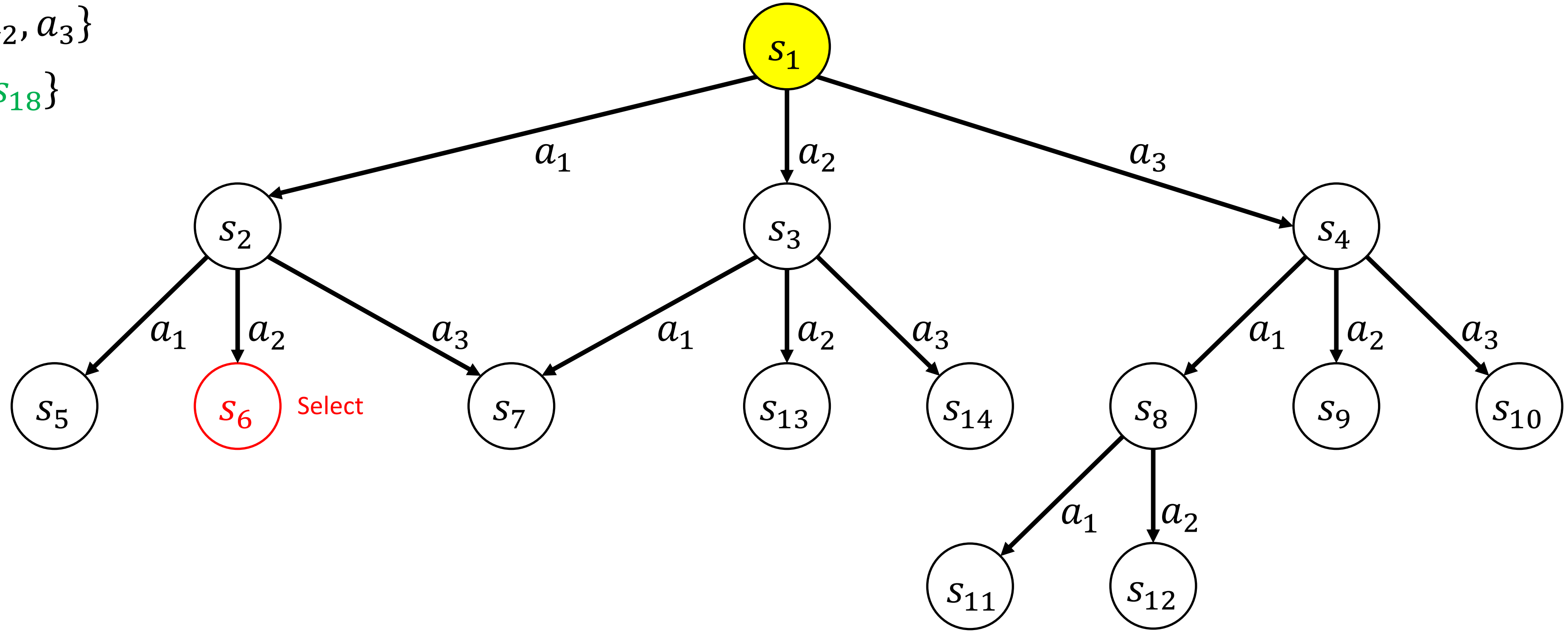
Classical Planning

Technique: A* State-Space Search (12)

$\mathcal{S} = \{s_1, s_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

$G = \{s_{17}, s_{18}\}$



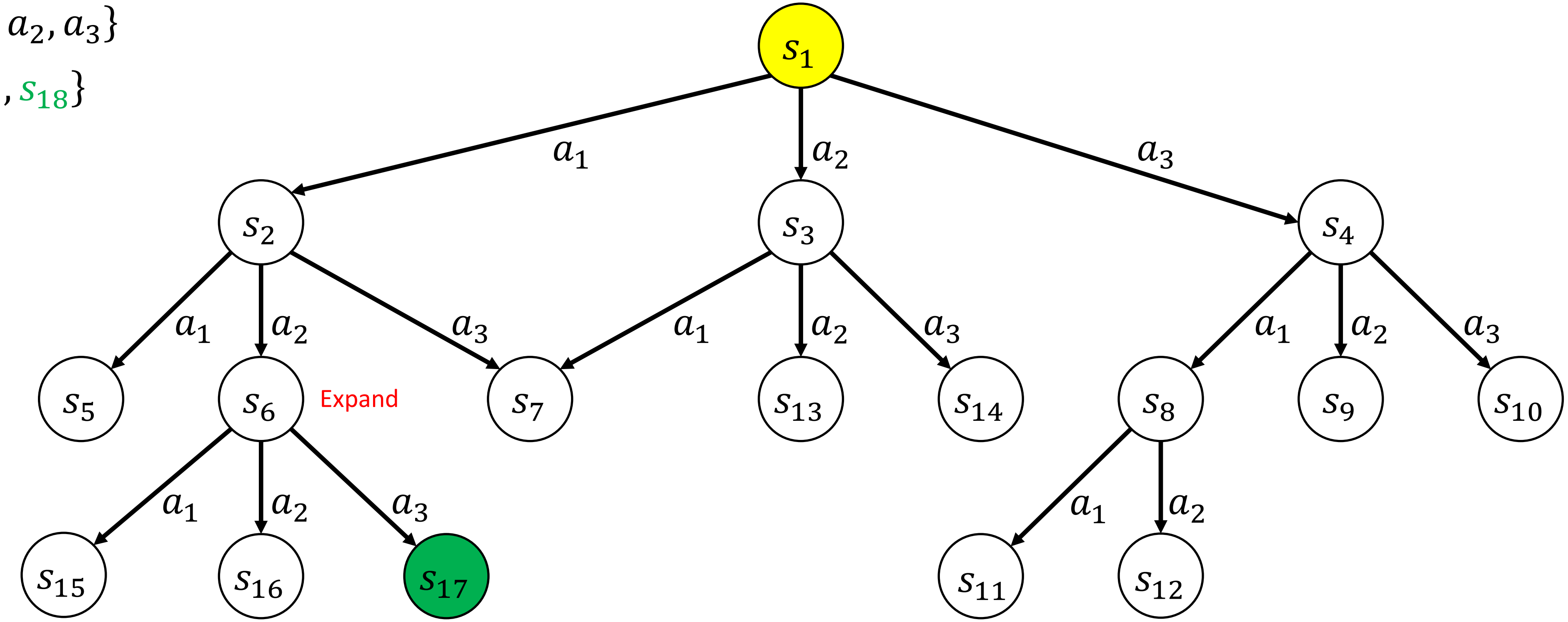
Classical Planning

Technique: A* State-Space Search (13)

$\mathcal{S} = \{s_1, s_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

$G = \{s_{17}, s_{18}\}$



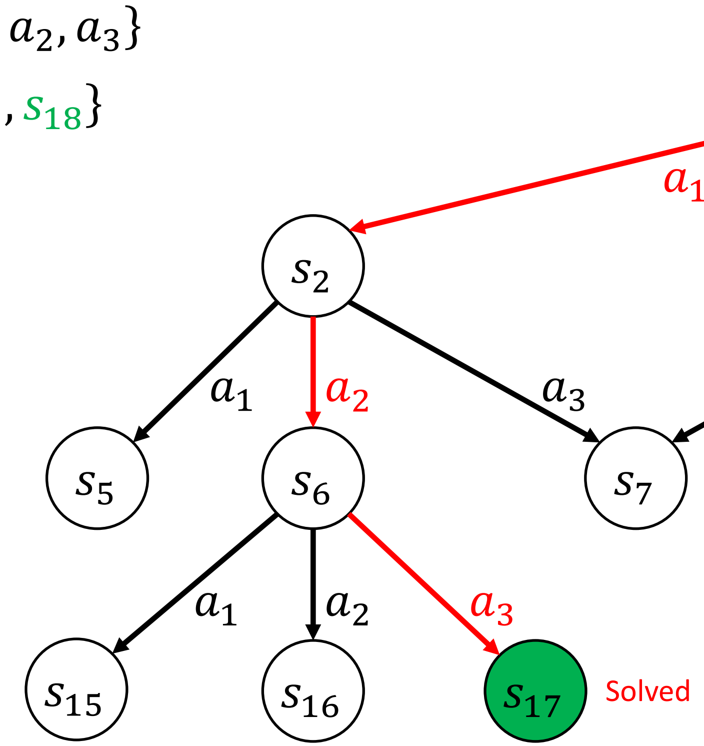
Classical Planning

Technique: A* State-Space Search (14)

$\mathcal{S} = \{s_1, s_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

$G = \{s_{17}, s_{18}\}$



Plan $\pi = (a_1, a_2, a_3)$

Classical Planning

Pros & Cons

- - Generally not reflective of the real-world
  - In practice, plans may be sub-optimal or incomplete (e.g. if, in reality, actions are not actually deterministic)

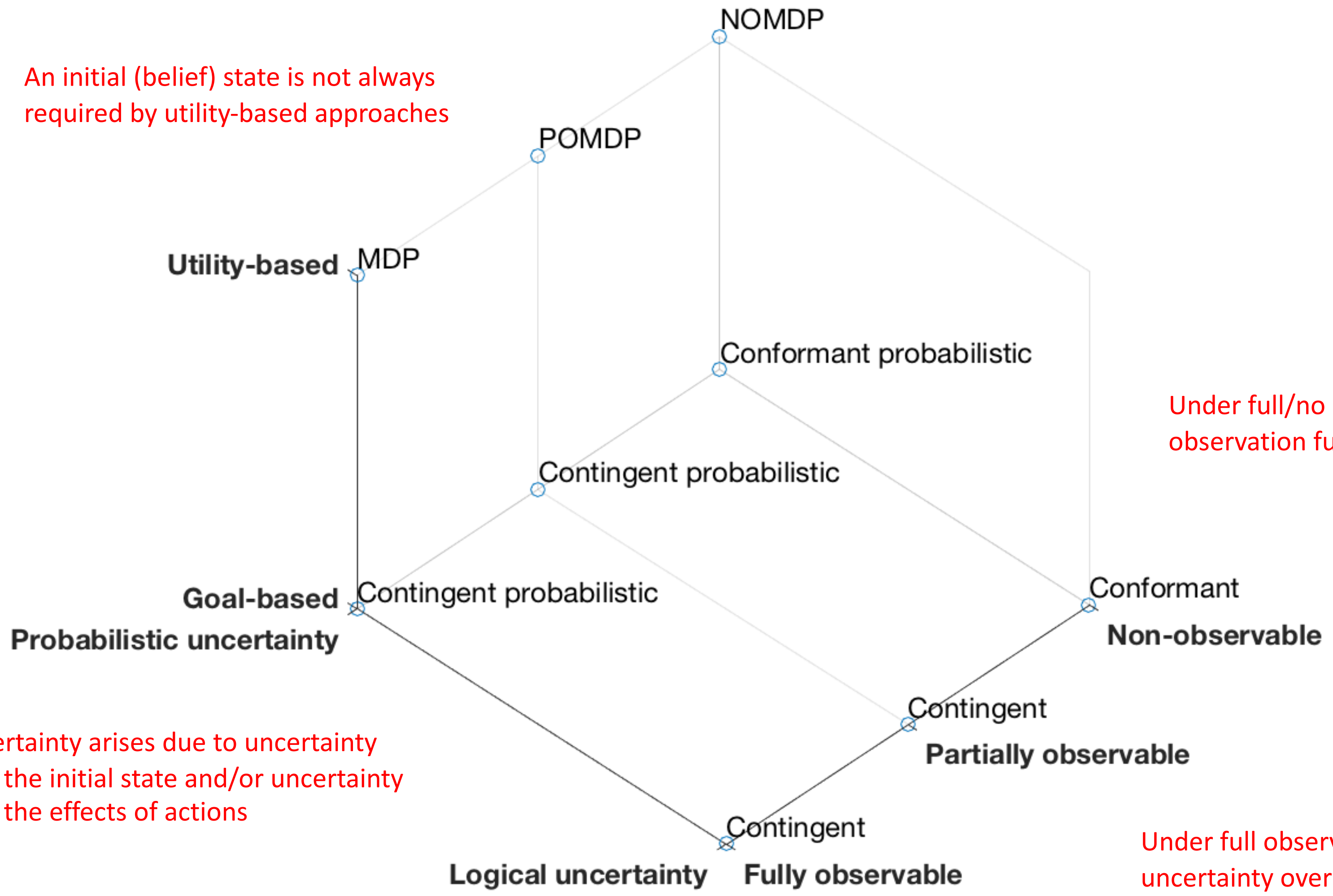
- + Comparatively easy to describe
  - Standard PDDL action descriptions are fairly compact and intuitive

- + State-of-the-art classical planners (e.g. FF) can scale to very large/complex problems
  - Benefit of making strong (unrealistic?) assumptions about properties of the task environment

- + Richer planning problems can be "translated" into classical planning
  - Arguably, classical planning remains an active area of research largely for this reason

Planning Under Uncertainty

An initial (belief) state is not always required by utility-based approaches



Under full/no observability, an observation function is not required

Uncertainty arises due to uncertainty over the initial state and/or uncertainty over the effects of actions

Under full observability, there is no uncertainty over the initial state

Conformant Planning

Definition

- Conformant planning problem $(\mathcal{S}, \mathcal{A}, b_1, T, G)$
  - Initial belief state $b_1 \in B$ where $B = 2^{\mathcal{S}}$
  - Deterministic transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ or non-deterministic transition function $T : \mathcal{S} \times \mathcal{A} \to 2^{\mathcal{S}}$
  - Set of goal states $G \subseteq \mathcal{S}$

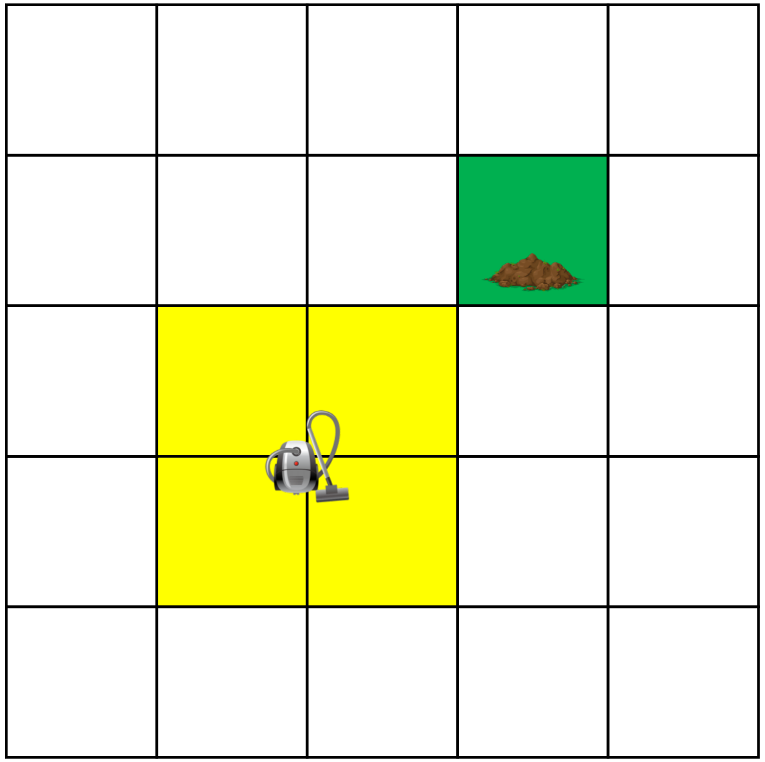
- A plan is a finite sequence of actions
  - Goal is satisfied when $b \subseteq G$
  - Must work for any possible initial state and/or successor state without relying on observations

- T is extended to belief states
  - $T(b, a) = \{T(s, a) \mid s \in b\}$ if T is deterministic
  - $T(b, a) = \{s' \in T(s, a) \mid s \in b\}$ if T is non-deterministic

Conformant Planning

Example (1)

$\pi = (\ldots)$

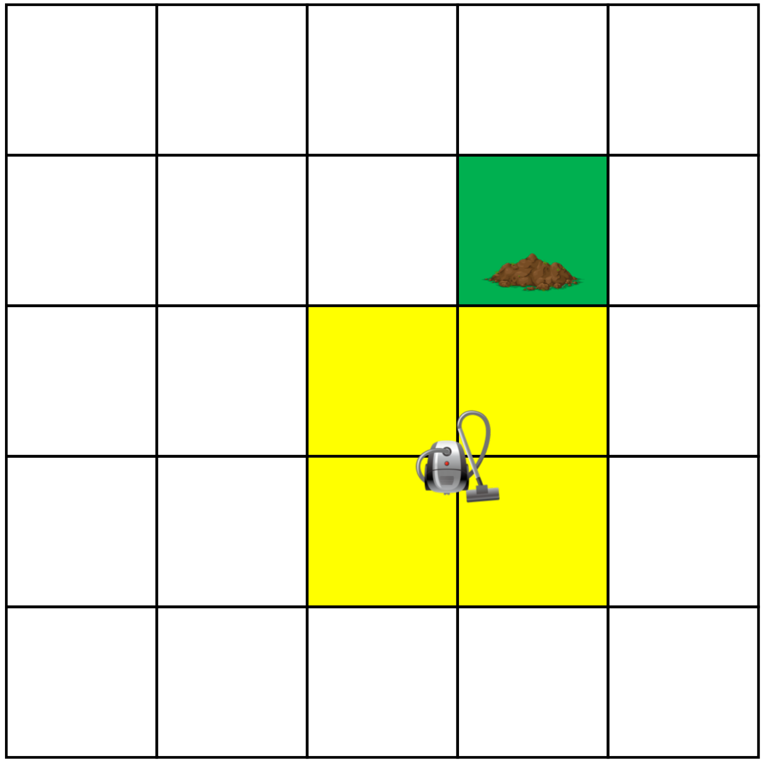


Execute Move(Right)

Conformant Planning

Example (2)

$\pi = (\text{Move}(\text{Right}), \ldots)$

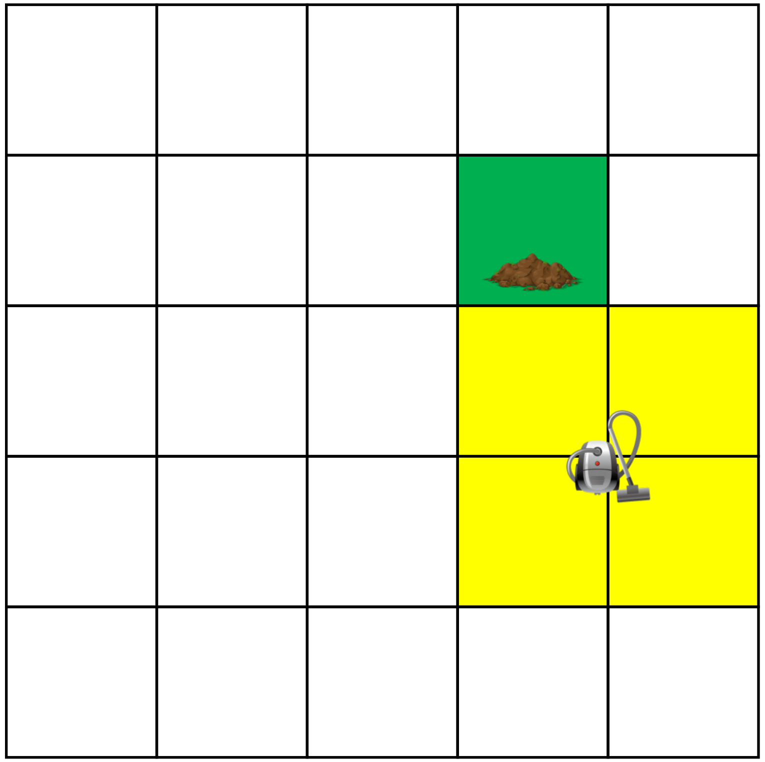


Execute Move(Right)

Conformant Planning

Example (3)

$\pi = (\text{Move}(\text{Right}), \text{Move}(\text{Right}), \dots)$

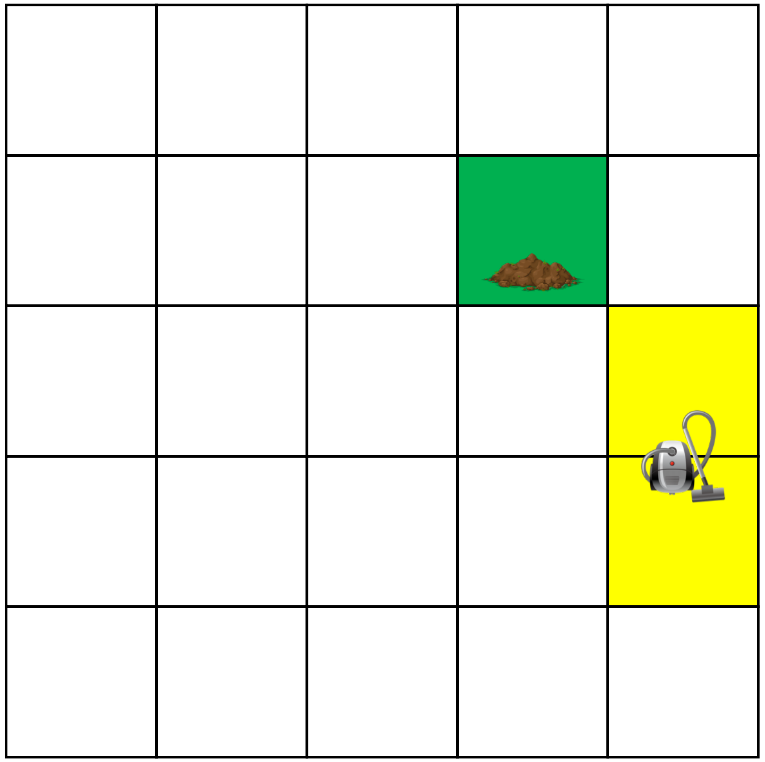


Execute $\text{Move}(\text{Right})$

Conformant Planning

Example (4)

$\pi = (\text{Move(Right)}, \text{Move(Right)}, \text{Move(Right)}, \ldots)$

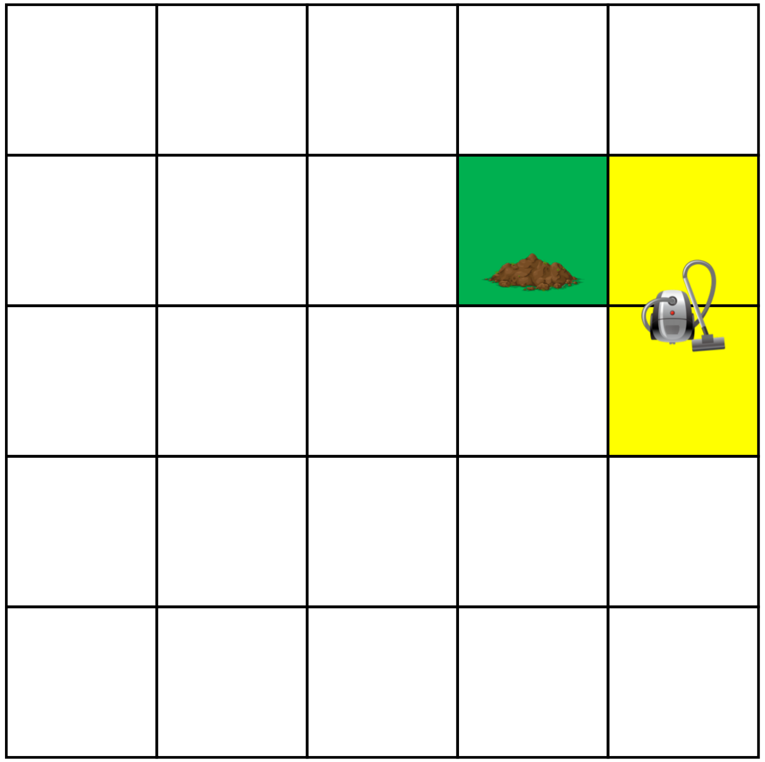


Execute Move(Up)

Conformant Planning

Example (5)

$\pi = (\mathrm{Move(Right)}, \mathrm{Move(Right)}, \mathrm{Move(Right)}, \mathrm{Move(Up)}, \ldots)$

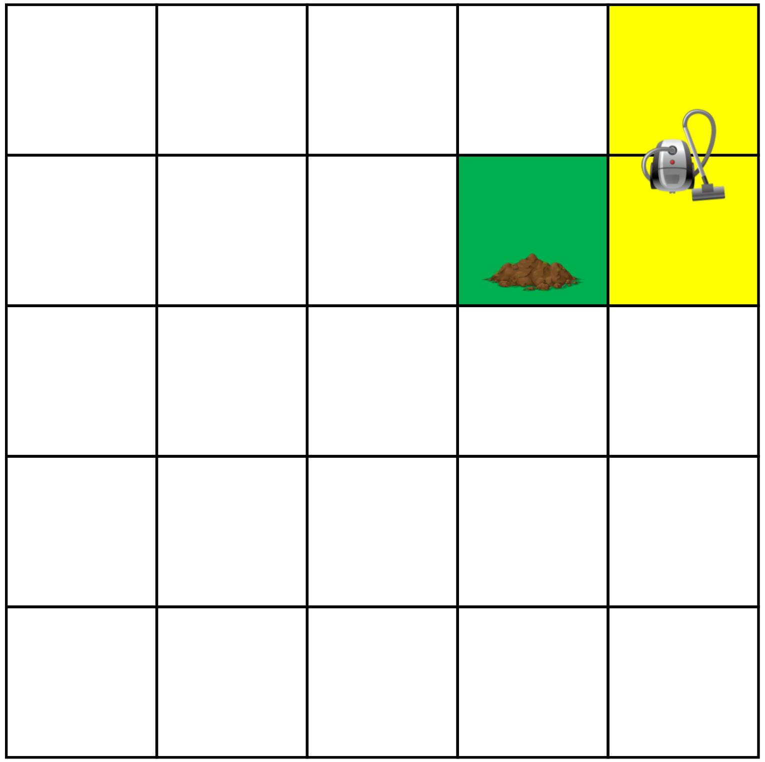


Execute Move(Up)

Conformant Planning

Example (6)

$\pi = (\text{Move(Right)}, \text{Move(Right)}, \text{Move(Right)}, \text{Move(Up)}, \textcolor{red}{\text{Move(Up)}}, \ldots)$

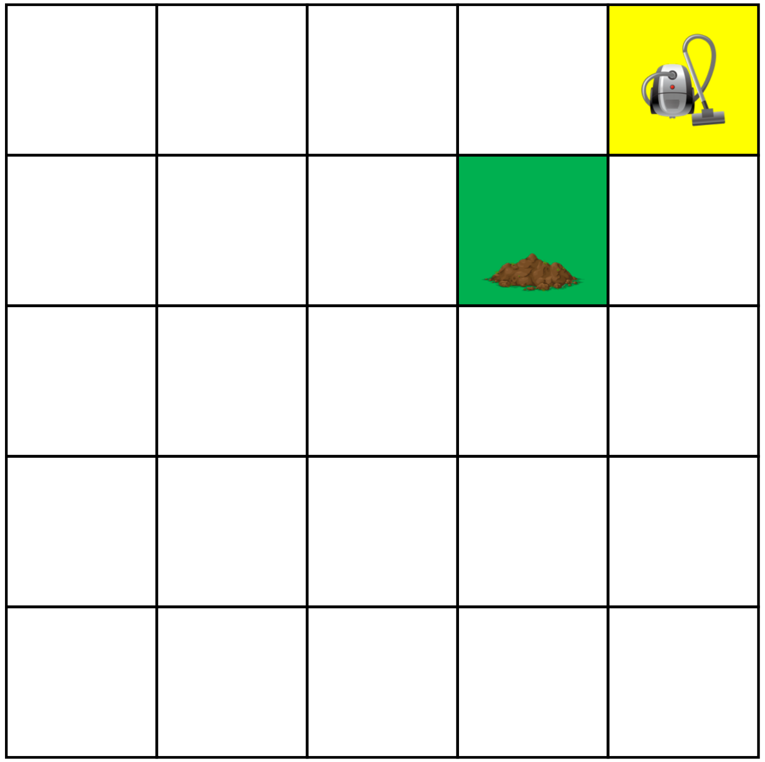


Execute Move(Up)

Conformant Planning

Example (7)

$\pi = (\text{Move}(\text{Right}), \text{Move}(\text{Right}), \text{Move}(\text{Right}), \text{Move}(\text{Up}), \text{Move}(\text{Up}), \text{Move}(\text{Up}), \ldots)$

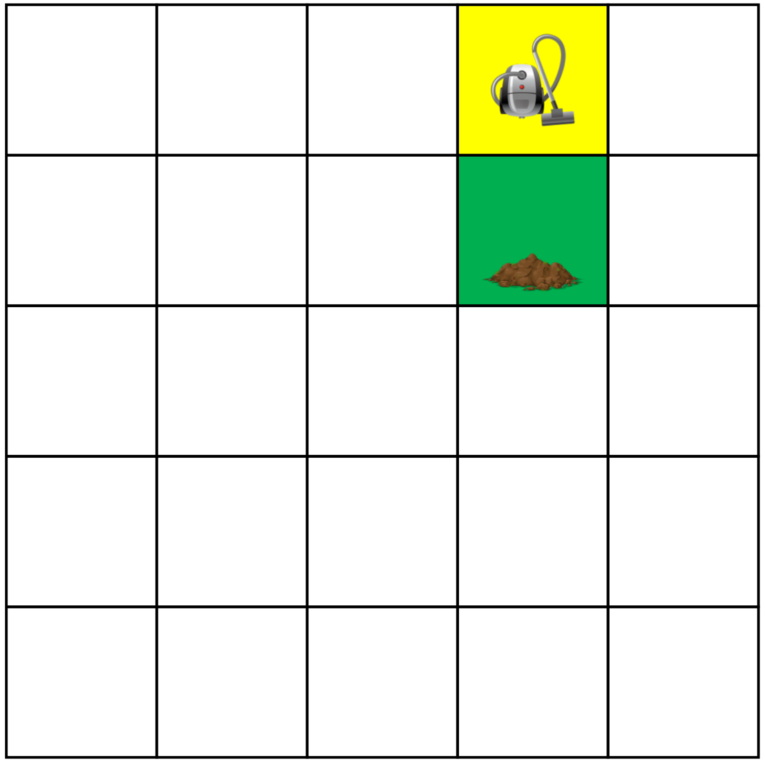


Execute $\text{Move}(\text{Left})$

Conformant Planning

Example (8)

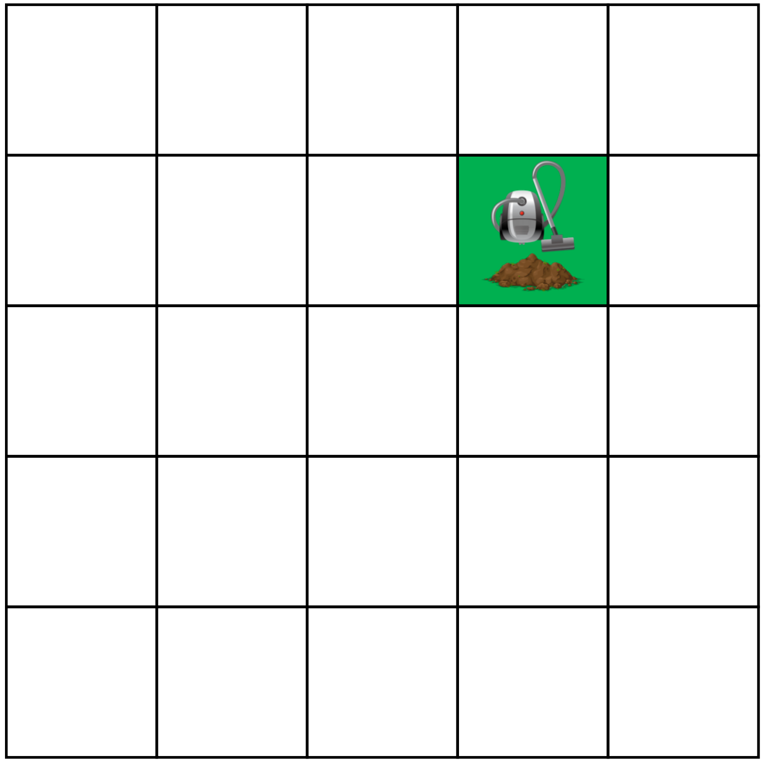
$\pi = (\text{Move(Right)}, \text{Move(Right)}, \text{Move(Right)}, \text{Move(Up)}, \text{Move(Up)}, \text{Move(Up)}, \text{Move(Left)}, \ldots)$



Execute Move(Down)

Conformant Planning
Example (9)

$\pi = (\text{Move(Right)}, \text{Move(Right)}, \text{Move(Right)}, \text{Move(Up)}, \text{Move(Up)}, \text{Move(Up)}, \text{Move(Left)}, \text{Move(Down)})$



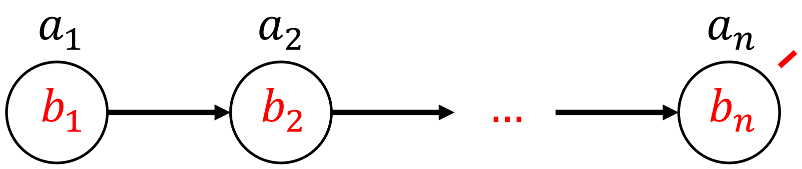
Goal achieved

Conformant Planning

Solution Form

1. Finite sequence of actions $p = (a_1, a_2, \dots, a_n)$ from initial belief state $b_1 \in B$

Goal $T(b_n, a_n) \subseteq G$



3. Policy graph (B', E, b_1, π_B)
   - Directed edges $E \subseteq B' \times B'$
   - Rooted at b_1
   - Nodes labelled by π_B

Successor belief state $T(b_1, a_1) = b_2$

2. Partial belief-based policy $\pi_B : B' \rightarrow \mathcal{A}$ with $B' \subseteq B$ such that π_B is closed with respect to $b_1 \in B$
   - If $b \in B$ is reachable from b_1 via π_B and T, then π_B is closed with respect b_1 if and only if $b \in B'$

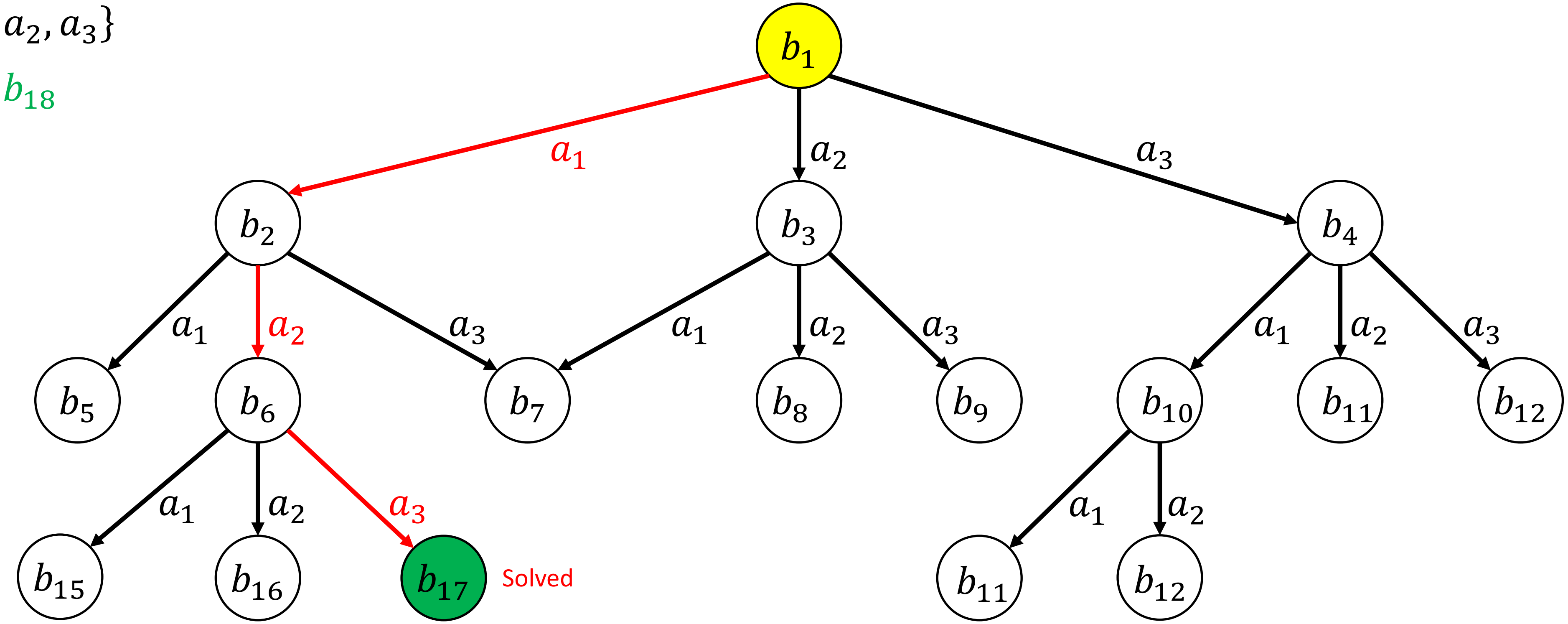
Conformant Planning

Technique: A* Belief-Space Search

$B = \{b_1, b_2, \dots\}$

$\mathcal{A} = \{a_1, a_2, a_3\}$

$G \supseteq b_{17}, b_{18}$



Plan $\pi = (a_1, a_2, a_3)$

Contingent Planning

Definition

- Contingent planning problem with full observability $(\mathcal{S}, \mathcal{A}, s_1, T, G)$
  - Initial state $s_1 \in \mathcal{S}$
  - Non-deterministic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$
  - Set of goal states $G \subseteq \mathcal{S}$

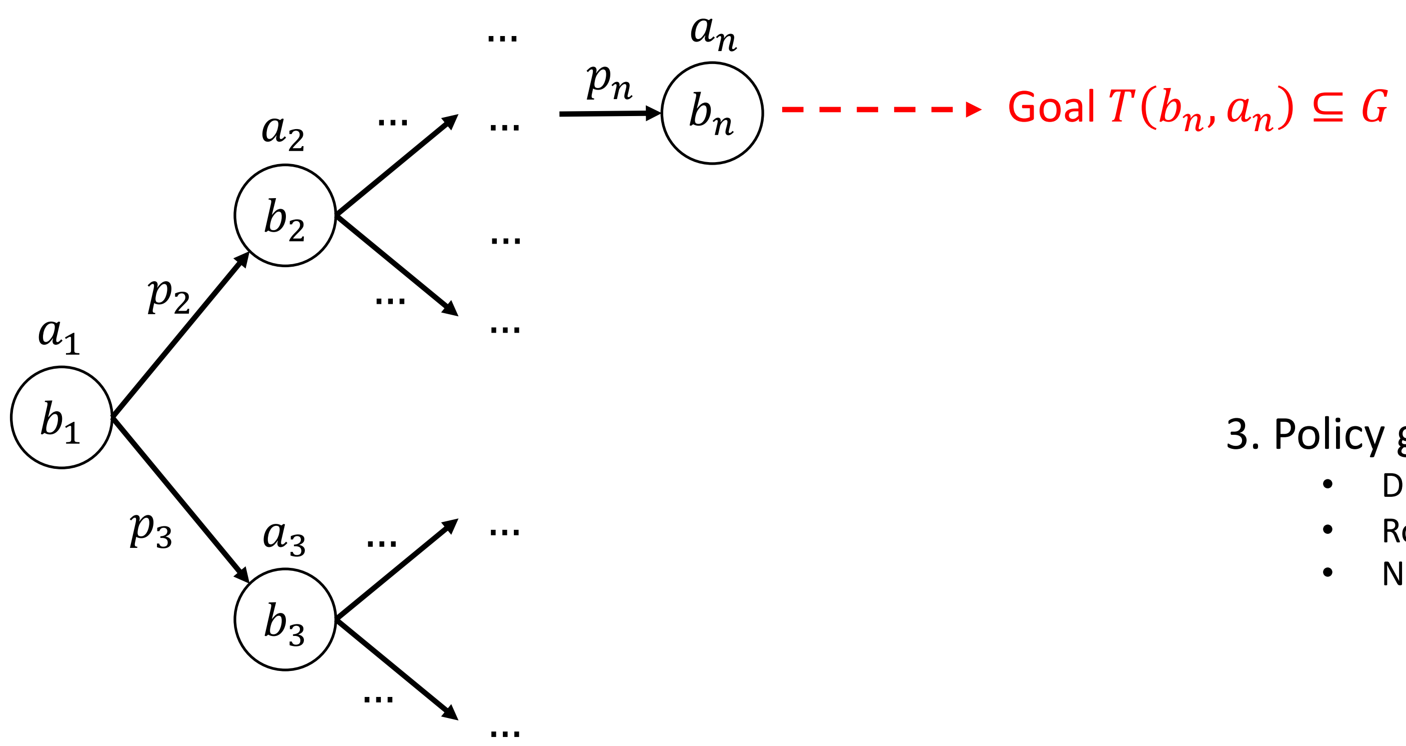
- Contingent planning problem with partial observability $(\mathcal{S}, \mathcal{A}, b_1, T, O, G)$
  - Initial belief state $b_1 \in B$ where $B = 2^{\mathcal{S}}$
  - Deterministic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ or non-deterministic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$
  - Deterministic observation function $O : \mathcal{S} \rightarrow \mathcal{P}$ or non-deterministic observation function $O : \mathcal{S} \rightarrow 2^{\mathcal{P}}$
  - Set of goal states $G \subseteq \mathcal{S}$

- A plan is a branching structure (e.g. a tree or graph)
  - Must work for any possible percept
  - May contain cycles, but sometimes an acyclic requirement is imposed

Contingent Planning

Solution Form

2. Partial belief-based policy $\pi_B : B' \to \mathcal{A}$ with $B' \subseteq B$ such that π_B is closed with respect to $b_1 \in B$



3. Policy graph $(B', \mathcal{P}, E, b_1, \pi_B)$

- Directed multi-edges $E \subseteq B' \times \mathcal{P} \times B'$
- Rooted at b_1
- Nodes labelled by π_B

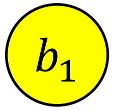
Contingent Planning

Technique: AO* Belief-Space Search

$B = \{b_1, b_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2\}$

$G \supseteq b_{16}, b_{17}, b_{18}$



Initialise

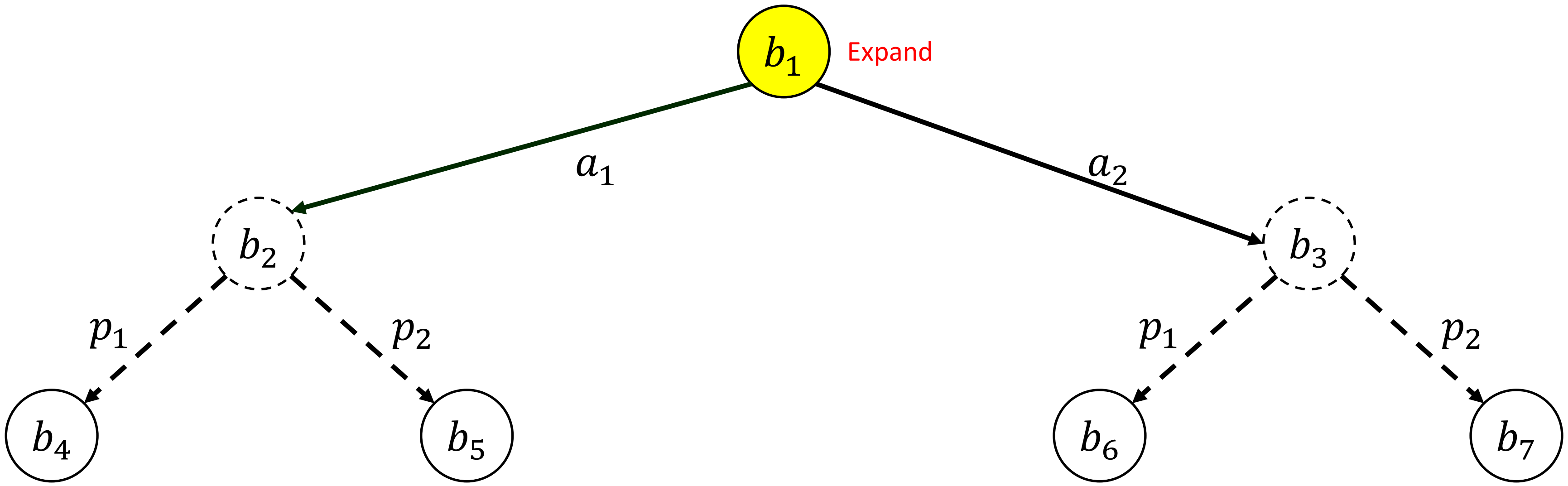
Contingent Planning

Technique: AO* Belief-Space Search

$B = \{b_1, b_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2\}$

$G \supseteq b_{16}, b_{17}, b_{18}$



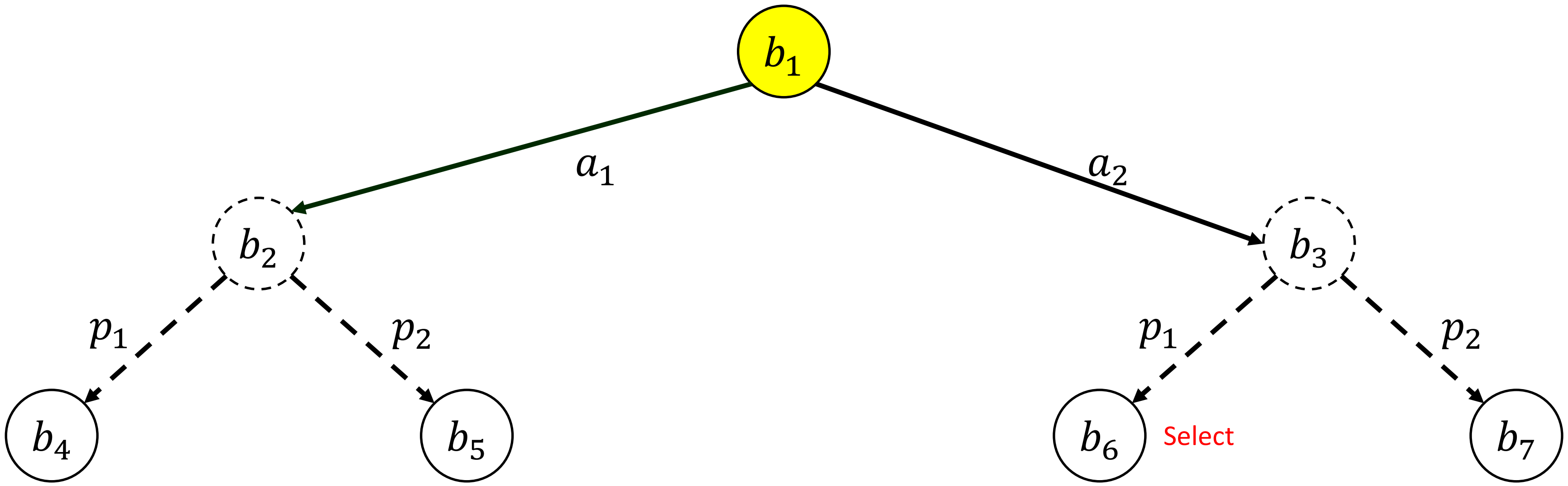
Contingent Planning

Technique: AO* Belief-Space Search

$B = \{b_1, b_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2\}$

$G \supseteq b_{16}, b_{17}, b_{18}$



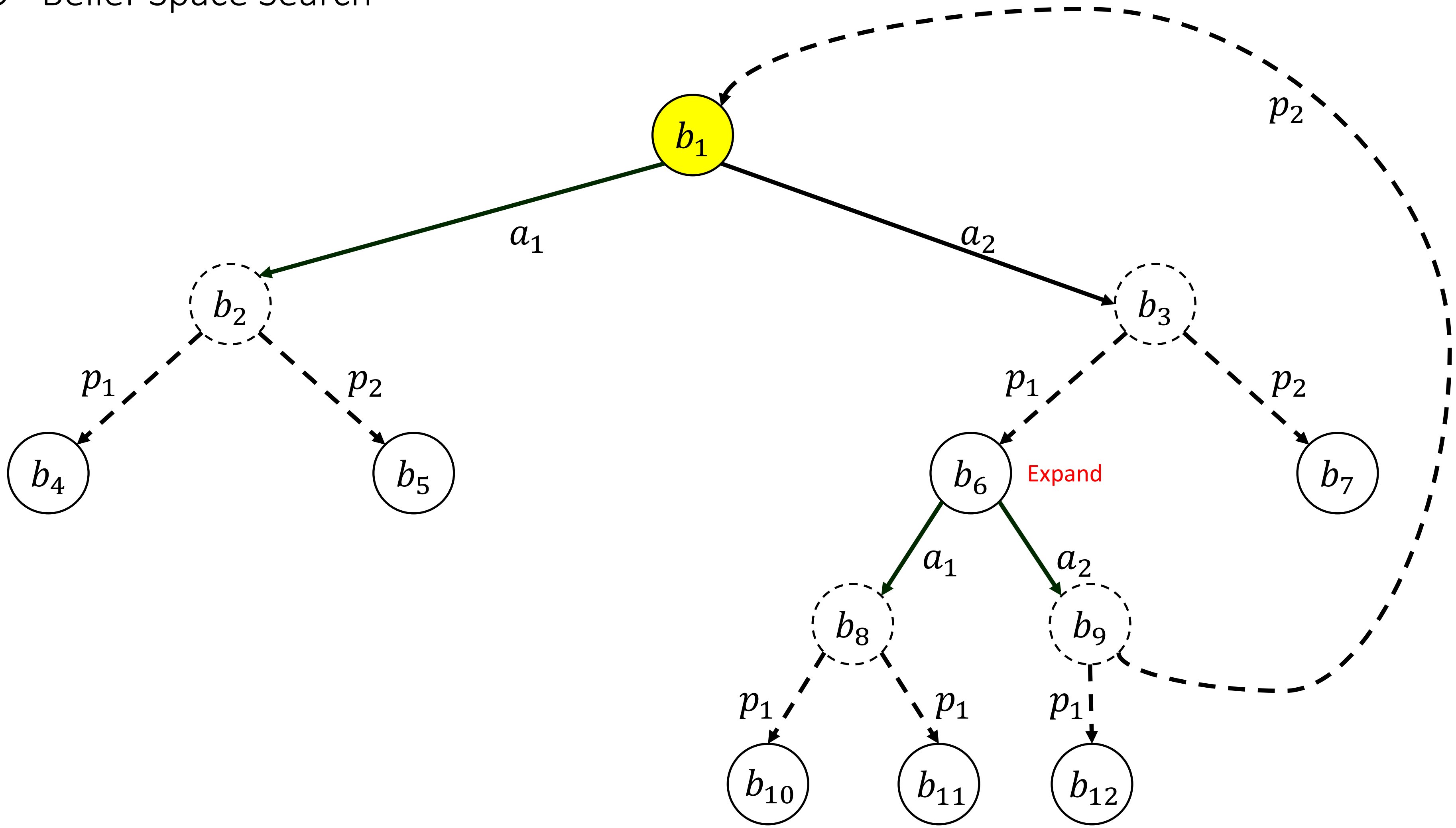
Contingent Planning

Technique: AO* Belief-Space Search

$B = \{b_1, b_2, \dots\}$

$\mathcal{A} = \{a_1, a_2\}$

$G \supseteq b_{16}, b_{17}, b_{18}$



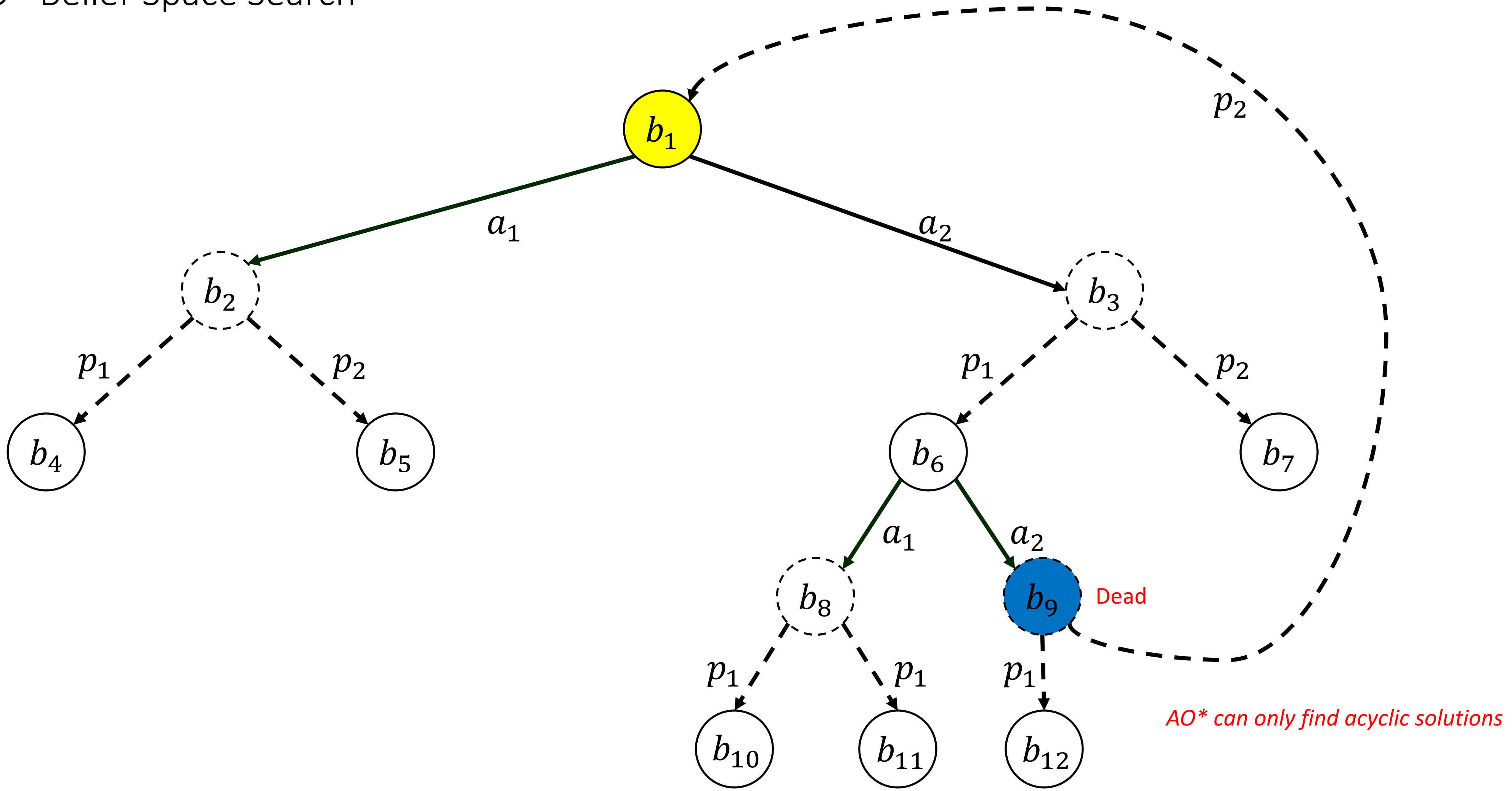
Contingent Planning

Technique: AO* Belief-Space Search

$B = \{b_1, b_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2\}$

$G \supseteq b_{16}, b_{17}, b_{18}$



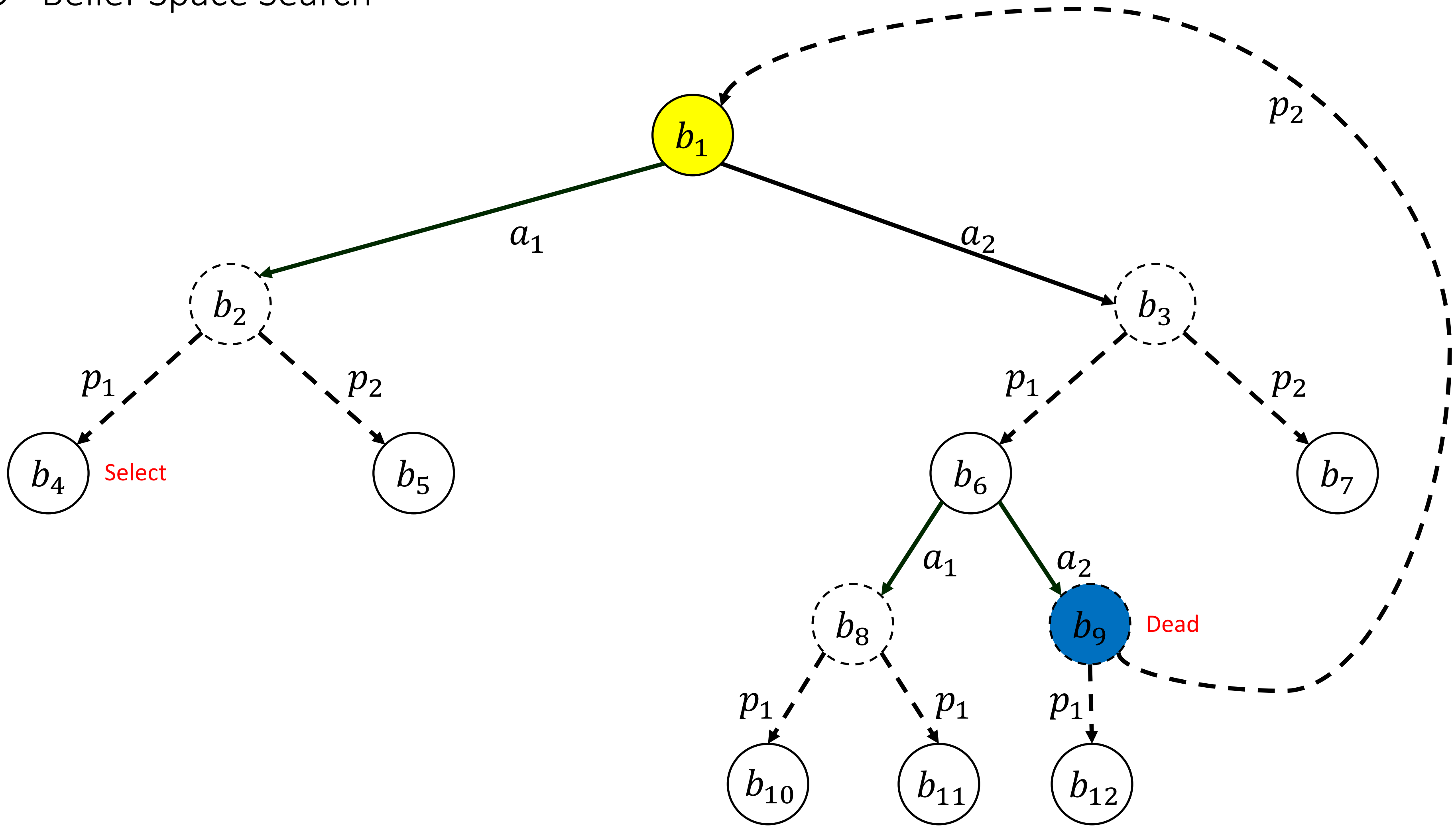
Contingent Planning

Technique: AO* Belief-Space Search

$B = \{b_1, b_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2\}$

$G \supseteq b_{16}, b_{17}, b_{18}$



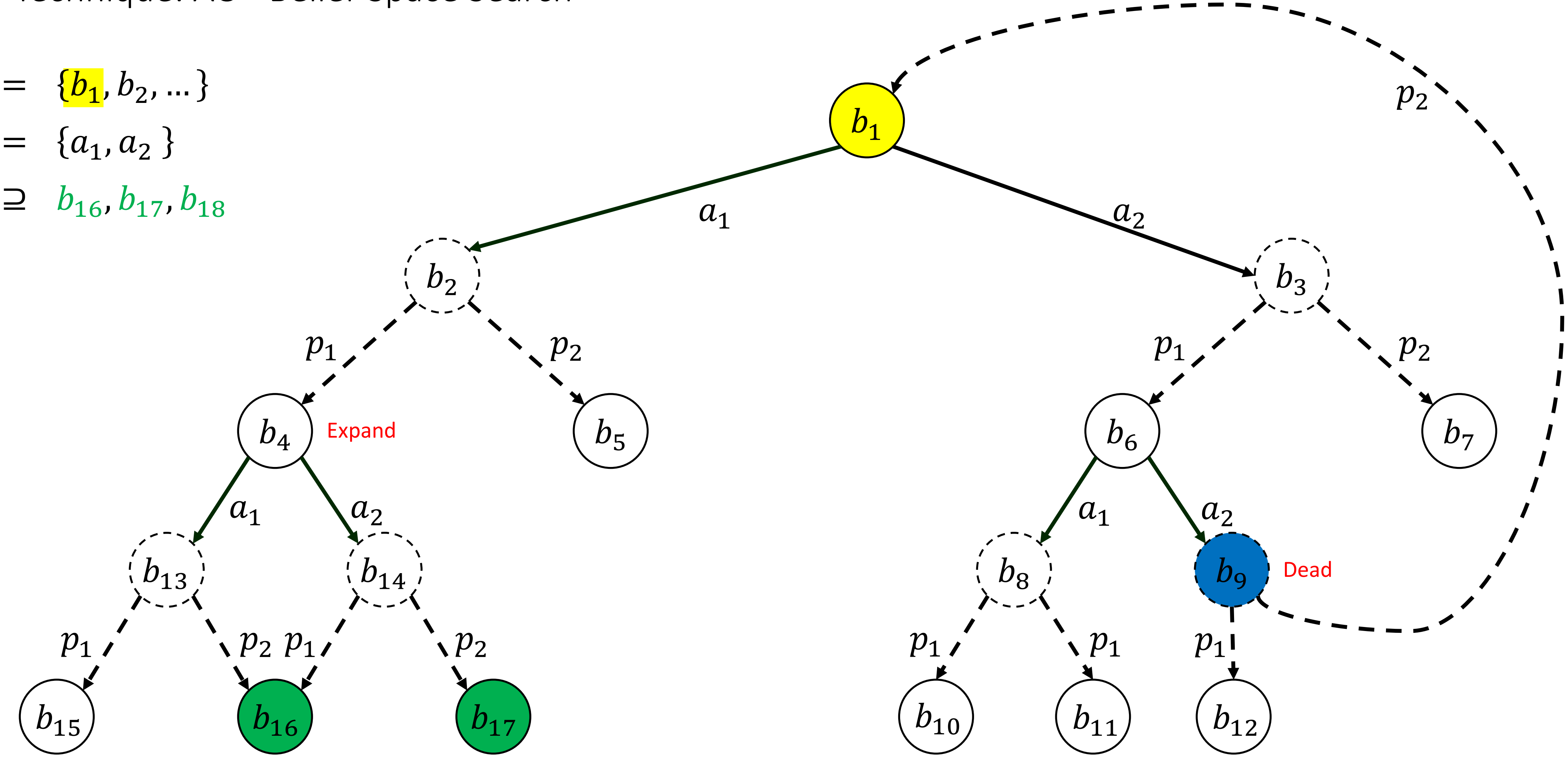
Contingent Planning

Technique: AO* Belief-Space Search

$B = \{b_1, b_2, \dots\}$

$\mathcal{A} = \{a_1, a_2\}$

$G \supseteq b_{16}, b_{17}, b_{18}$



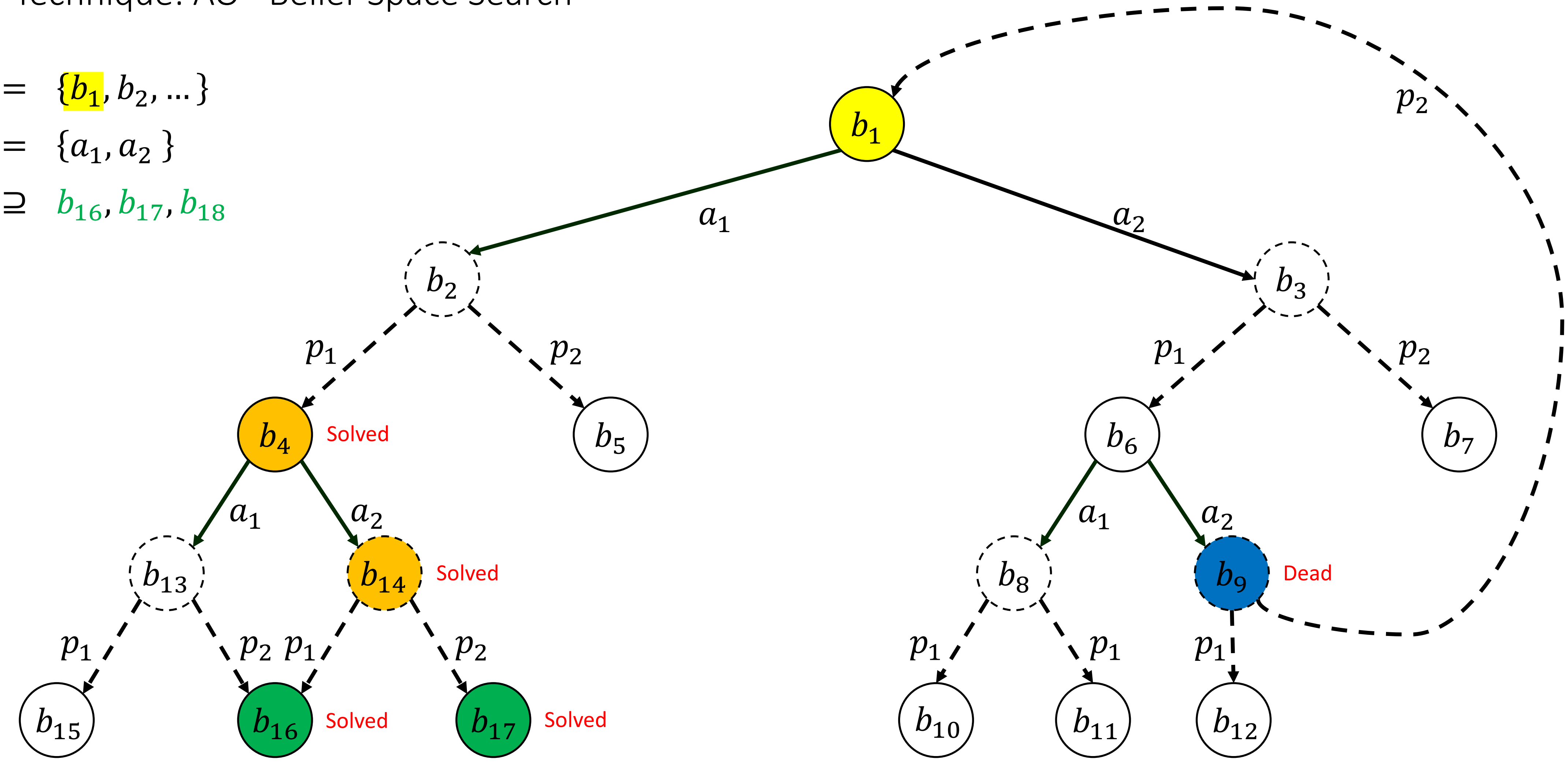
Contingent Planning

Technique: AO* Belief-Space Search

$$B = \{b_1, b_2, \dots\}$$

$$\mathcal{A} = \{a_1, a_2\}$$

$$G \supseteq b_{16}, b_{17}, b_{18}$$



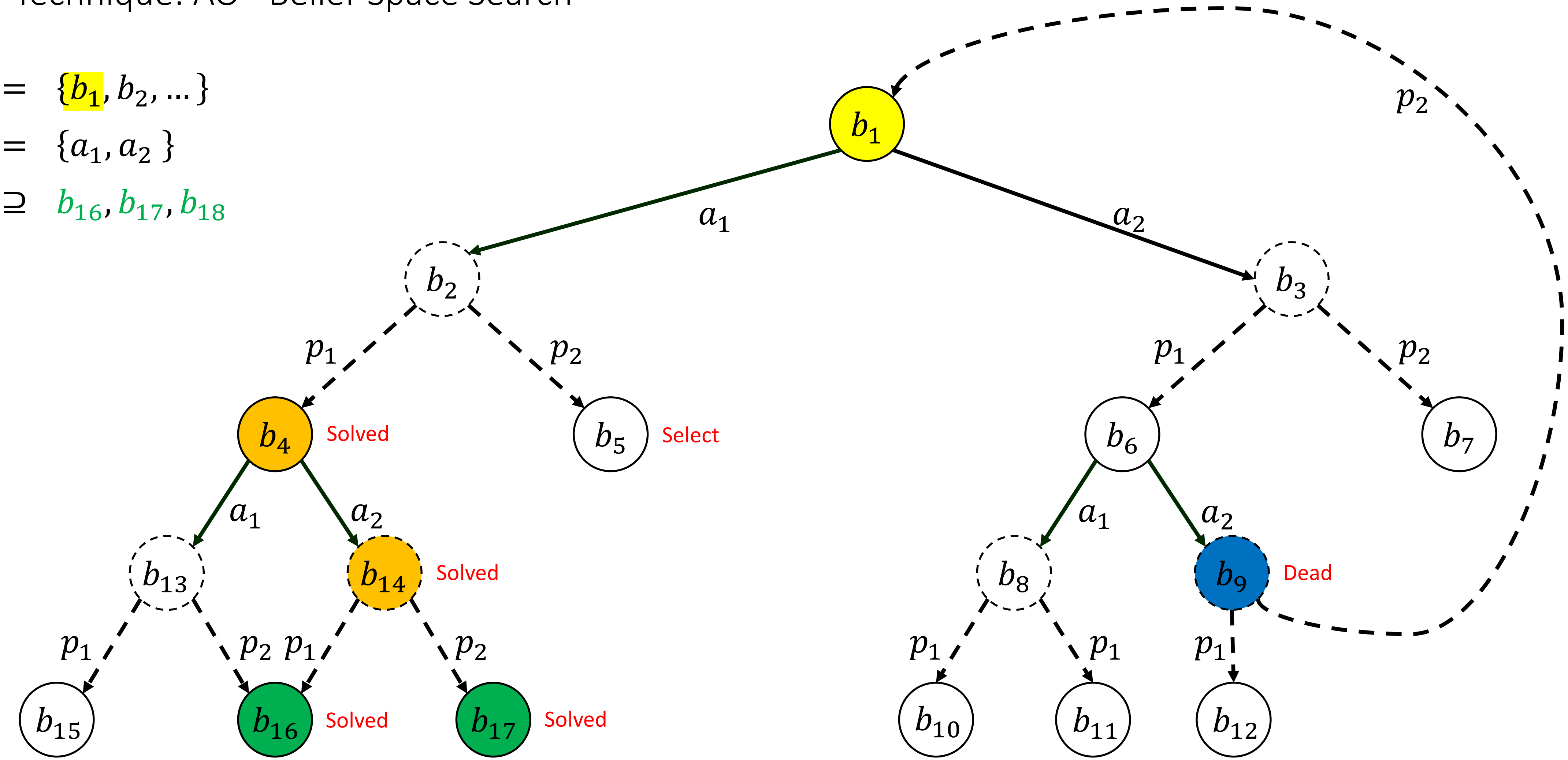
Contingent Planning

Technique: AO* Belief-Space Search

$B = \{b_1, b_2, \dots\}$

$\mathcal{A} = \{a_1, a_2\}$

$G \supseteq b_{16}, b_{17}, b_{18}$



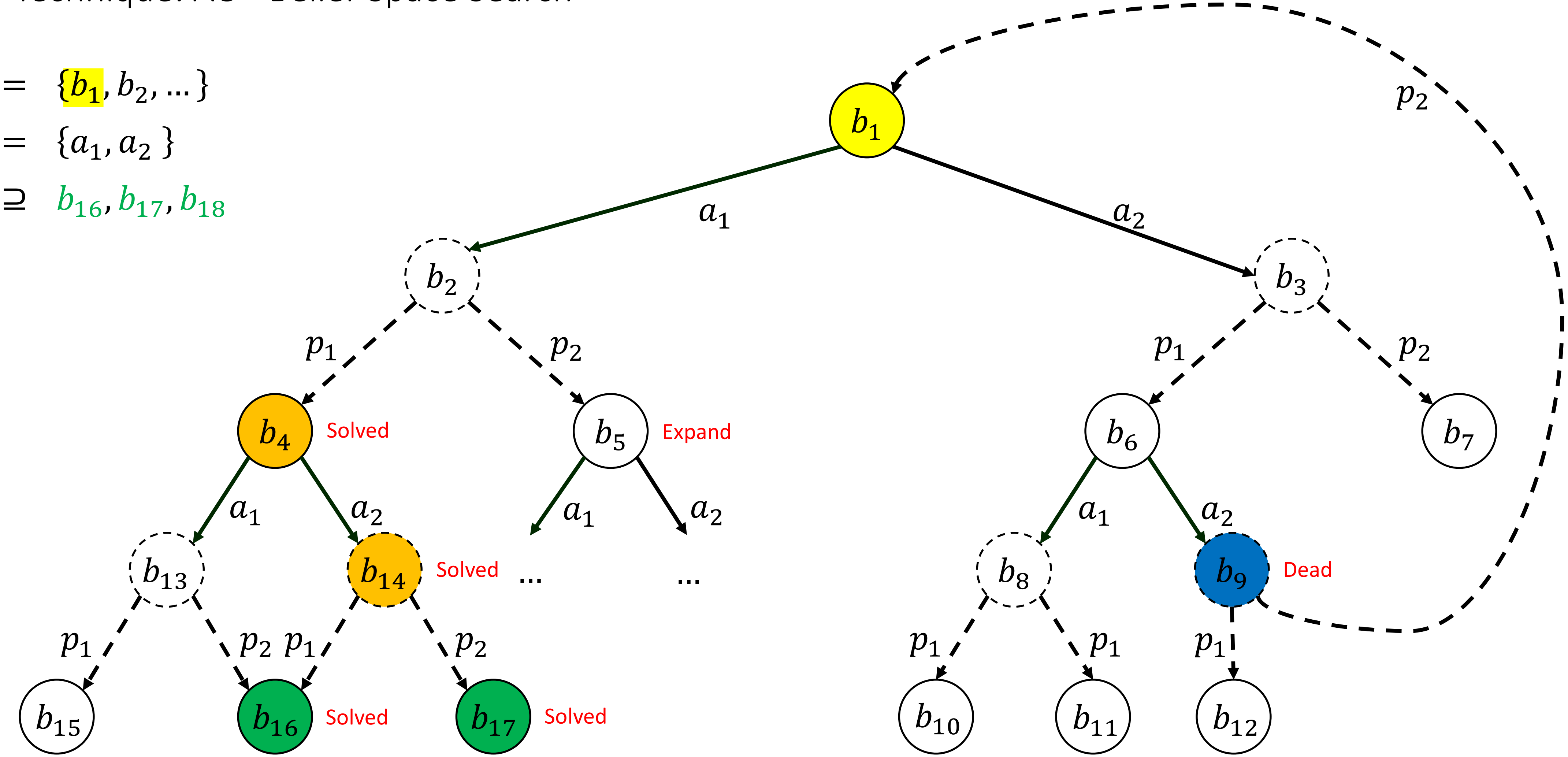
Contingent Planning

Technique: AO* Belief-Space Search

$\mathcal{B} = \{b_1, b_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2\}$

$G \supseteq b_{16}, b_{17}, b_{18}$



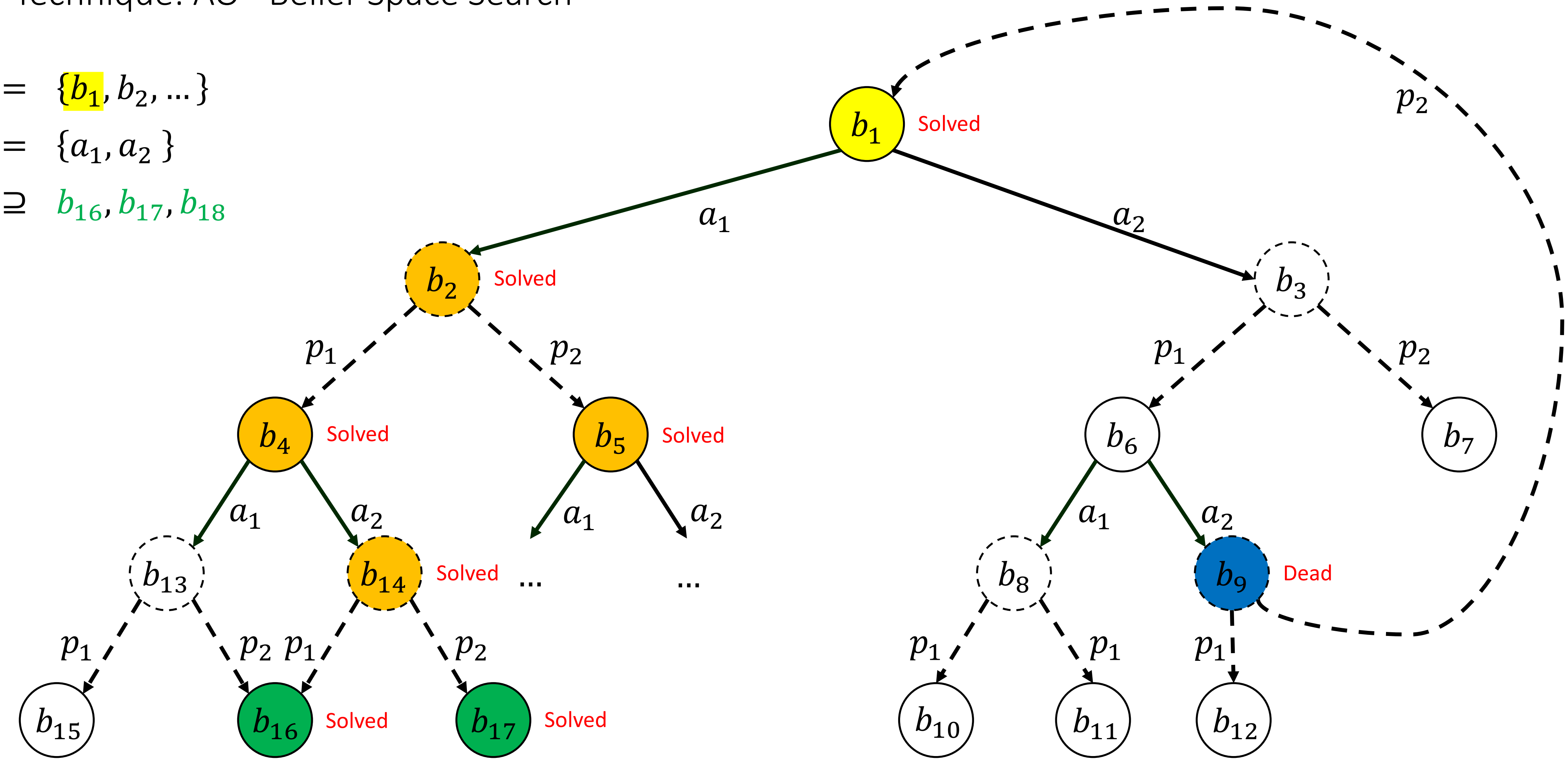
Contingent Planning

Technique: AO* Belief-Space Search

$B = \{b_1, b_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2\}$

$G \supseteq b_{16}, b_{17}, b_{18}$



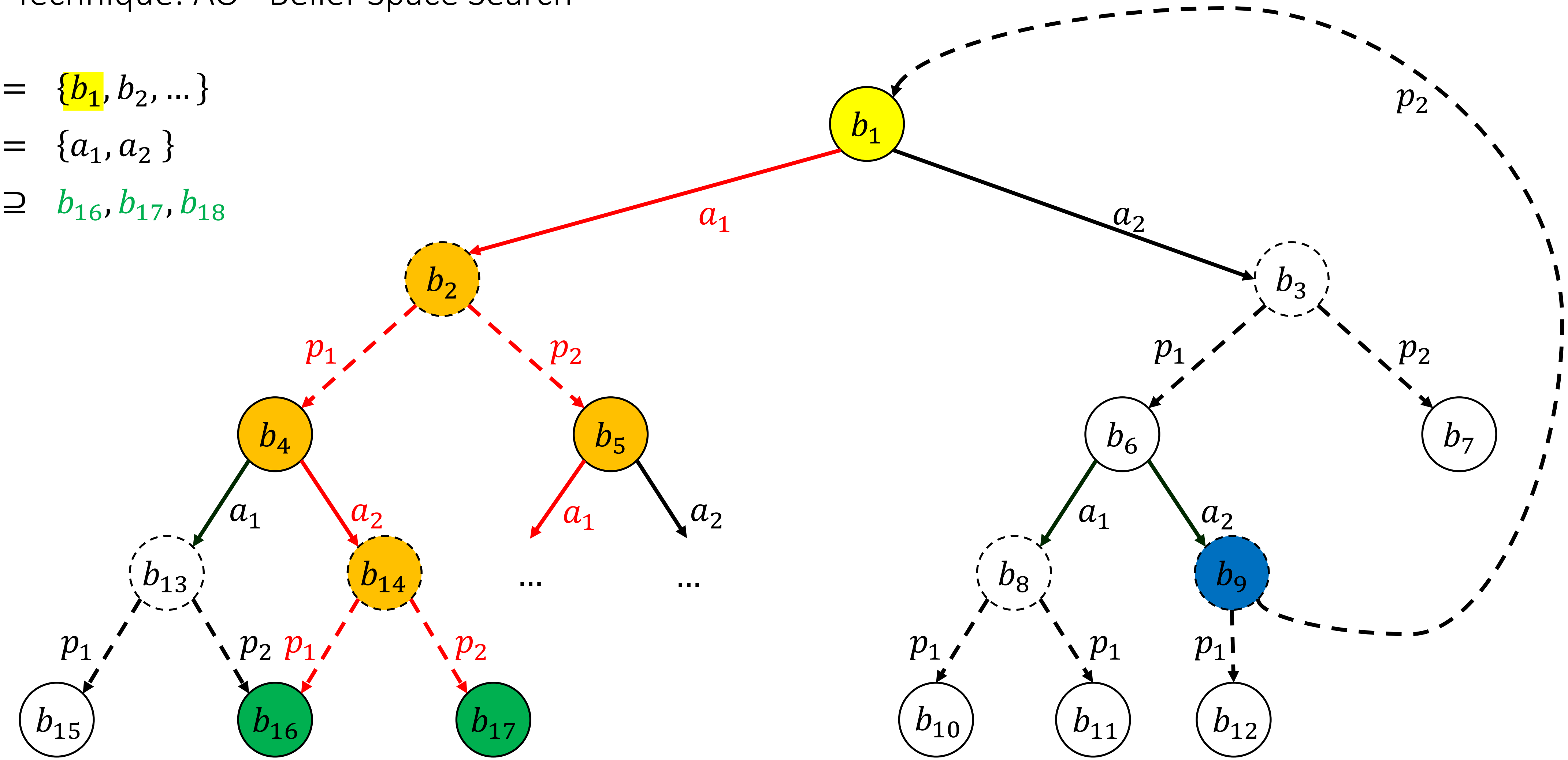
Contingent Planning

Technique: AO* Belief-Space Search

$B = \{b_1, b_2, \ldots\}$

$\mathcal{A} = \{a_1, a_2\}$

$G \supseteq b_{16}, b_{17}, b_{18}$



Markov Decision Process (MDP)

Definition (Fully Observable)

- Finite-horizon MDP $(\mathcal{S}, \mathcal{A}, T, R, h)$
  - Stochastic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$
  - Utility function $R : \mathcal{S} \rightarrow \mathbb{R}$
  - Horizon $h \in \mathbb{N}$

Equivalent definitions for partially observable MDPs (POMDPs) and non-observable MDPs (NOMDPs)

- Infinite-horizon, discounted-reward MDP $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$
  - Stochastic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$
  - Utility function $R : \mathcal{S} \rightarrow \mathbb{R}$
  - Discount factor $0 \leq \gamma < 1$

- Indefinite-horizon, goal-oriented MDP $(\mathcal{S}, \mathcal{A}, s_1, T, G)$
  - Initial state $s_1 \in \mathcal{S}$
  - Stochastic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$
  - Set of goal states $G \subseteq \mathcal{S}$

Type of MDP	Type of planning
Goal-oriented MDP	Contingent probabilistic planning with full observability
Goal-oriented POMDP	Contingent probabilistic planning with partial observability
Goal-oriented NOMDP	Conformant probabilistic planning

Online Planning

- What we have discussed so far is known as offline planning
  - Generate a complete plan and then execute in full

- An alternative is online planning
  - Generate an incomplete policy, execute until it is necessary to plan again, and repeat
  - Special case: generate "next action", execute, and repeat
  - An online planner is essentially an agent, since it must run in an environment and execute actions

- Incomplete policy
  - A partial state- or belief-based policy π is incomplete if π is not closed with respect to initial (belief) state $b_1 \in B$

- Plan-act procedure
  1. Execute $\pi_B : B' \rightarrow \mathcal{A}$ until some belief state $b \in B \setminus B'$ is encountered
  2. Update π_B to $\pi_B{}'$ via planning such that $\pi'_B : B'' \rightarrow \mathcal{A}$ with $B'' \supset B'$
  3. Return to 1.

Online Planning

Technique: Classical Replanning

Algorithm: Classical replanning

Full observability

Input: Goal-oriented MDP $(\mathcal{S}, \mathcal{A}, s_1, T, G)$

$s \leftarrow s_1$

for each $s' \in \mathcal{S}$ **do**

  for each $a \in \mathcal{A}$ **do**

    $T'(s', a) \leftarrow \operatorname{argmax}_{s'' \in \mathcal{S}} T(s', a, s'')$ `// most-probable outcome`

Many other determinization methods (e.g. all outcomes with costs as inverse transition probabilities)

while $s \notin G$ **do**

  if $\pi(s) =$ *undefined* **then**

    $\pi' \leftarrow$ Classical-Plan$(\mathcal{S}, \mathcal{A}, s, T', G)$ `// replan`

    $\pi \leftarrow$ Update(π, π')

  Execute$(\pi(s))$

  $s \leftarrow$ Sense$()$

Automated Planning & Agents

Some Reflections

- A plan/policy can be thought of as an agent function
  - See panel discussion on BDI vs. MDPs at MSDM'15 workshop @ AAMAS'15: "encode everything in an MDP"
  - A plan/planner provides a very limited view of agents (e.g. no multi-tasking, no exogenous events, no complex reasoning)

- Agent programs are compatible with automated planning
  - Plans can be generated by an automated planner and then executed by an agent

- Potential for plan misuse
  - For example, interleaving the execution of multiple classical plans may fail due to conflicting action-effects
  - Representing a plan as a policy helps to avoid this issue

Automated Planning & Agents

Other Approaches

- Partial-order (or least-commitment) planning
  - Plan is a partially ordered set of actions – no convenient state- or belief-based policy representation(?)
  - Difficult to extend to richer planning problems – especially those with probabilistic uncertainty
- Hierarchical task networks (HTN)
  - Input is a set compound tasks rather than a set of actions
  - Aim is to find a valid decomposition of those tasks into (executable) actions
  - Many successful industrial applications
- BDI-style plan selection
  - Instantiation of lifted plans and decomposition of sub-goals
  - Similar to HTN

This Lecture…

1. Classical planning
   - Planning languages
2. Planning under uncertainty
   - Conformant planning
   - Contingent planning
   - Markov decision processes
3. Online planning
   - Classical replanning

Next Lecture…

1. Agent-oriented programming
   - PRS
   - AgentSpeak
2. AgentSpeak
   - Logic programming
   - Syntax
   - Semantics
   - Interpreter
   - Programming

Bibliography

- Michael Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, 2nd edition, 2009.

Suggested reading (Chapter 3, 4, 10, 11, and 17)

- Stuart J. Russell & Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2009.

Suggested reading

- Hector Geffner & Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool, 2013.

- Anand S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, pages 42-55, 1996.

- Rafael H. Bordini, Jomi Fred Hübner, & Michael Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley & Sons, 2007.