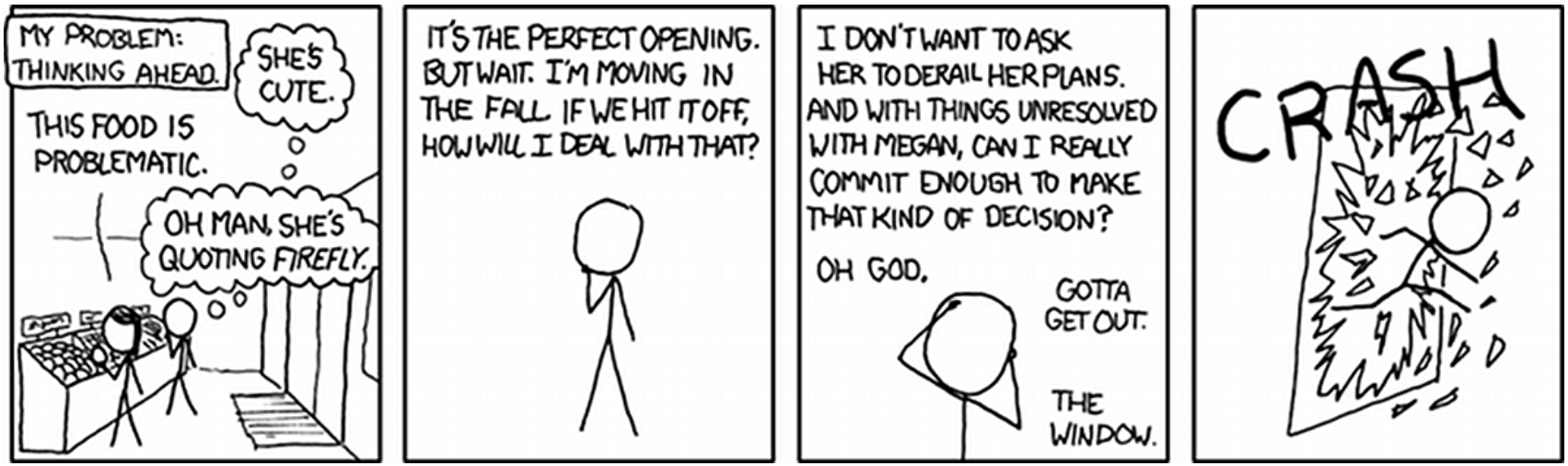
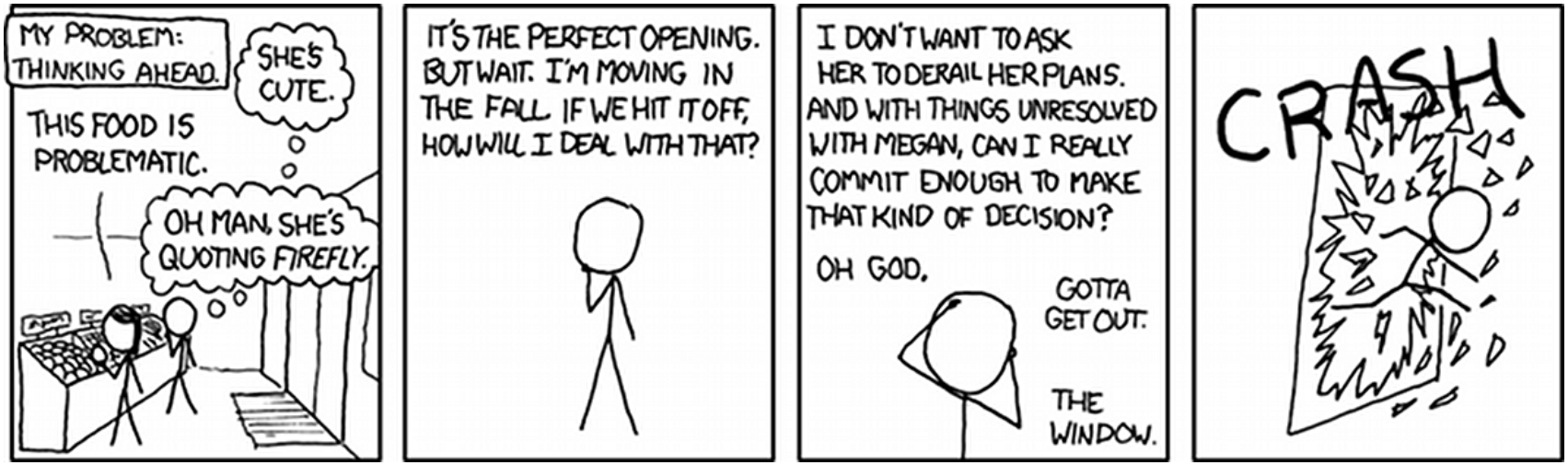
CS325 Artificial Intelligence
Chs. 10, 11 – Planning

Cengiz Günay, Emory Univ.

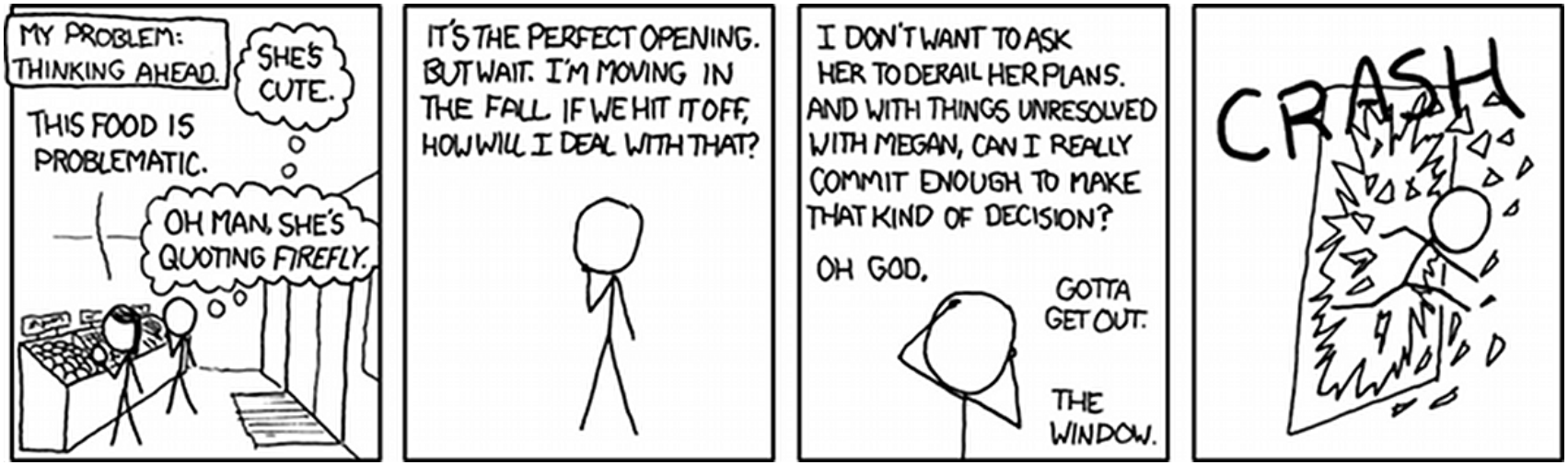


Spring 2013

Planning


MY PROBLEM: THINKING AHEAD.
SHE'S CUTE.
THIS FOOD IS PROBLEMATIC.
OH MAN, SHE'S QUOTING FIREFLY.
IT'S THE PERFECT OPENING. BUT WAIT. I'M MOVING IN THE FALL. IF WE HIT IT OFF, HOW WILL I DEAL WITH THAT?
I DON'T WANT TO ASK HER TO DERAIL HER PLANS. AND WITH THINGS UNRESOLVED WITH MEGAN, CAN I REALLY COMMIT ENOUGH TO MAKE THAT KIND OF DECISION?
OH GOD.
GOTTA GET OUT.
THE WINDOW.
CRASH

Planning



Planning is coming up with a solution for an agent:

- it's at the heart of AI

Entry/Exit Surveys

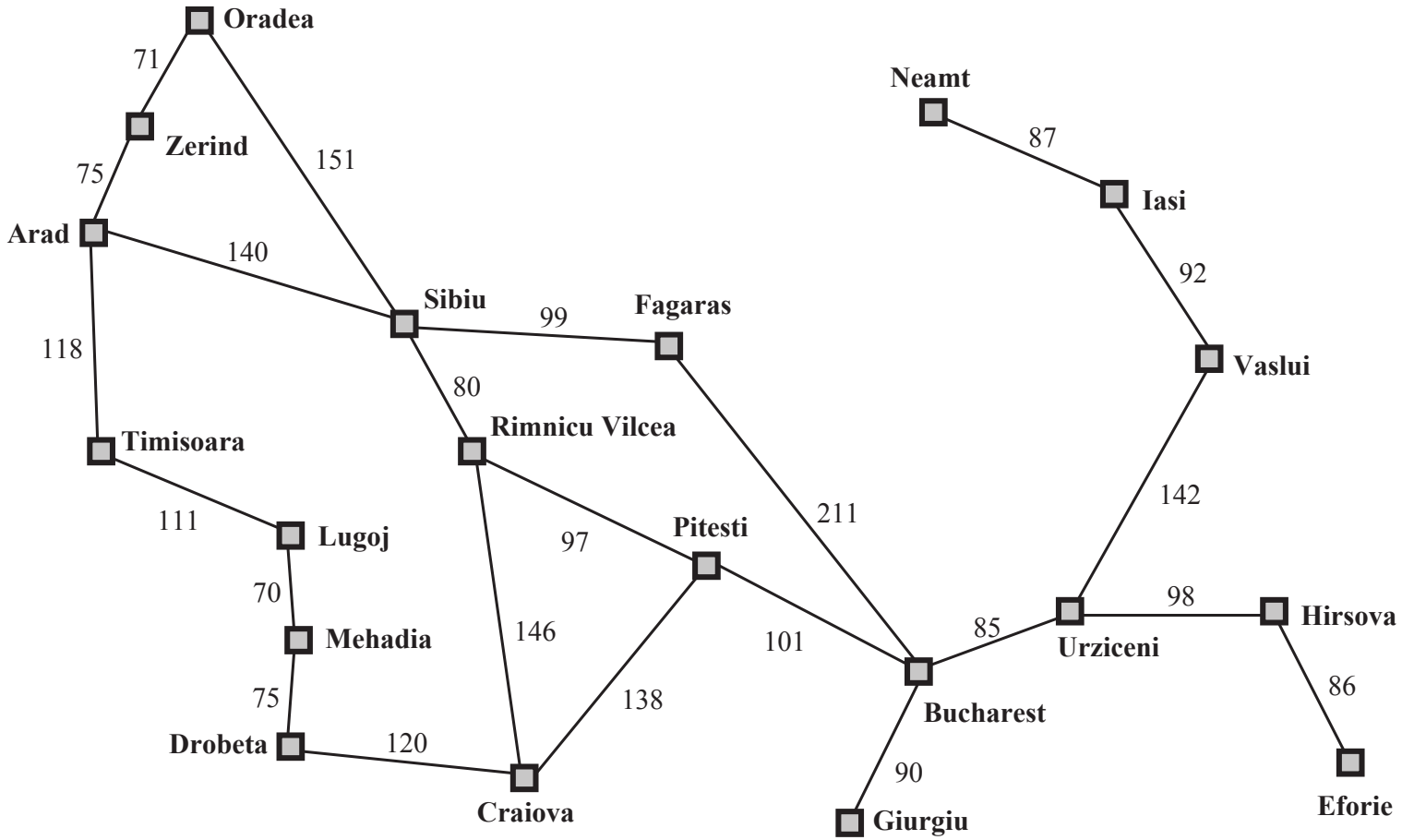
Exit survey: Knowledge Representation and Inference

- What knowledge representation do you think our brains have?
- What knowledge base topic/domain would you wish you had?

Entry survey: Planning (0.25 points of final grade)

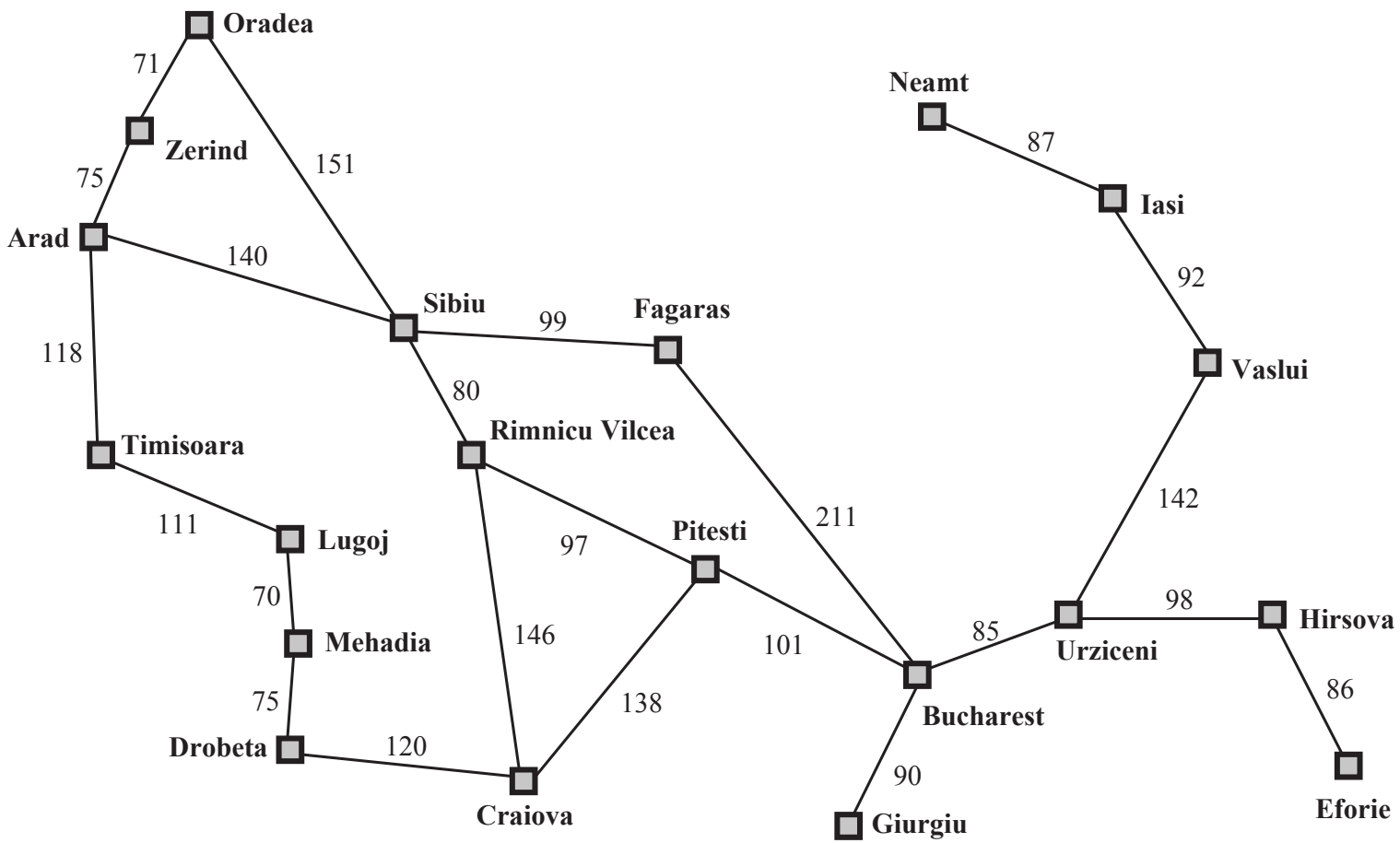
- What previous class topic would count as a planning algorithm?
- Where do you think you used an automated planning system?

Graph Search Is a Form of Planning



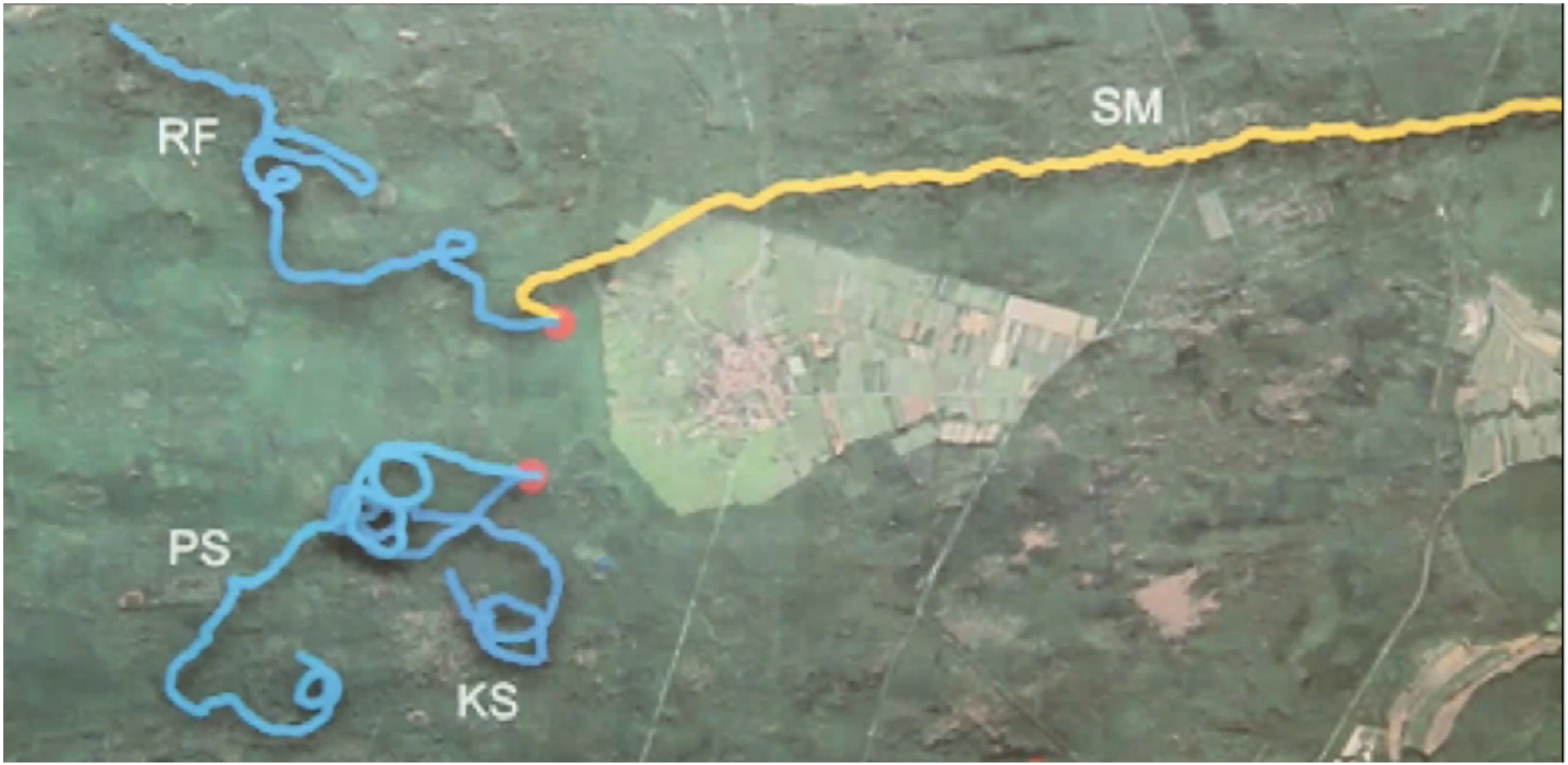
- Search algorithm like A^* is a planning method

Graph Search Is a Form of Planning

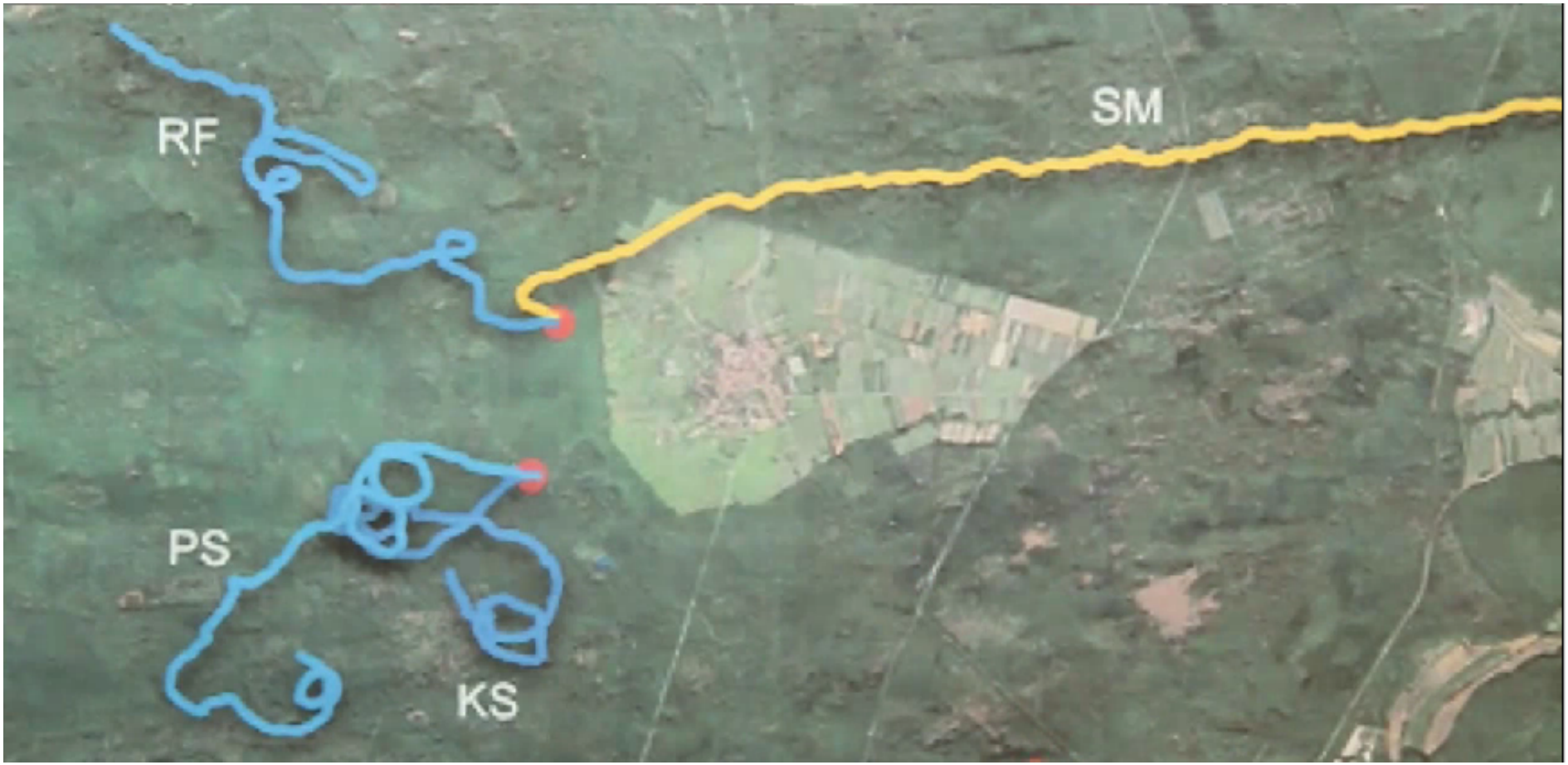


- Search algorithm like A^* is a planning method
- Only works for observable and deterministic environment
- What if you cannot plan all the way?

Blind Walking

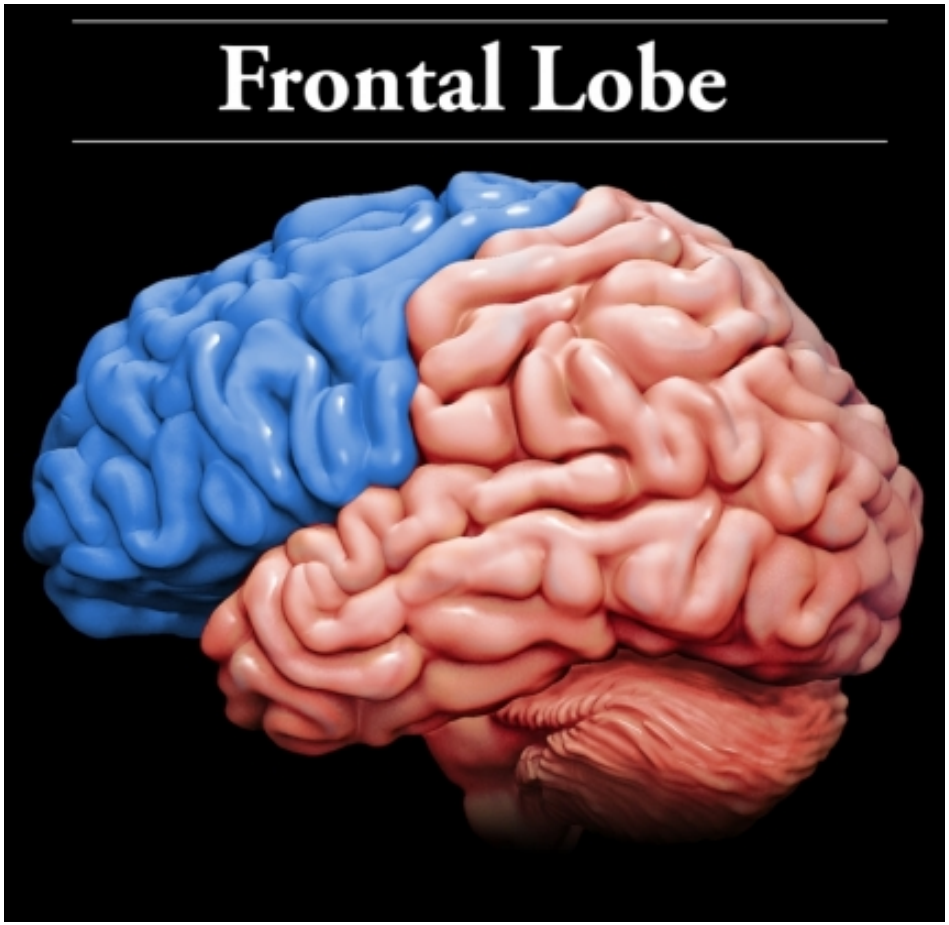


Blind Walking

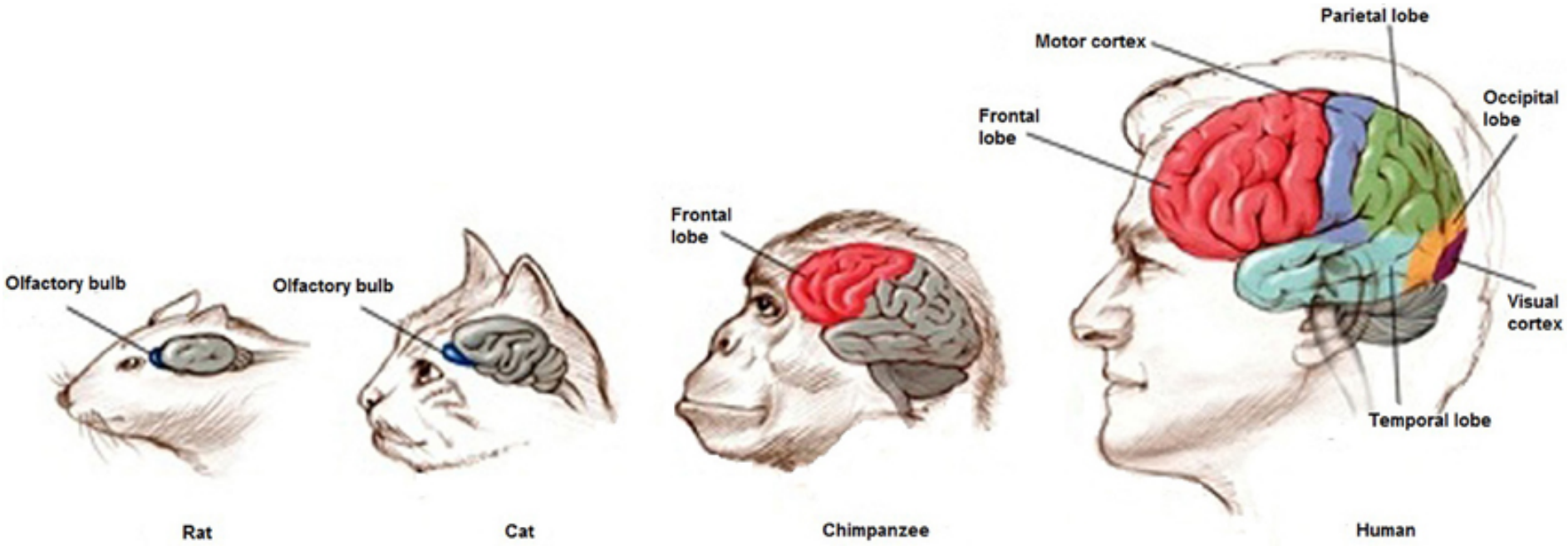


- Need to alternate plan & execution

How Does the Brain Do It?



How Does the Brain Do It?



- Frontal lobes are biggest in humans, dedicated to **planning, reasoning** and other high-order skills

So When Do You Need Planning?

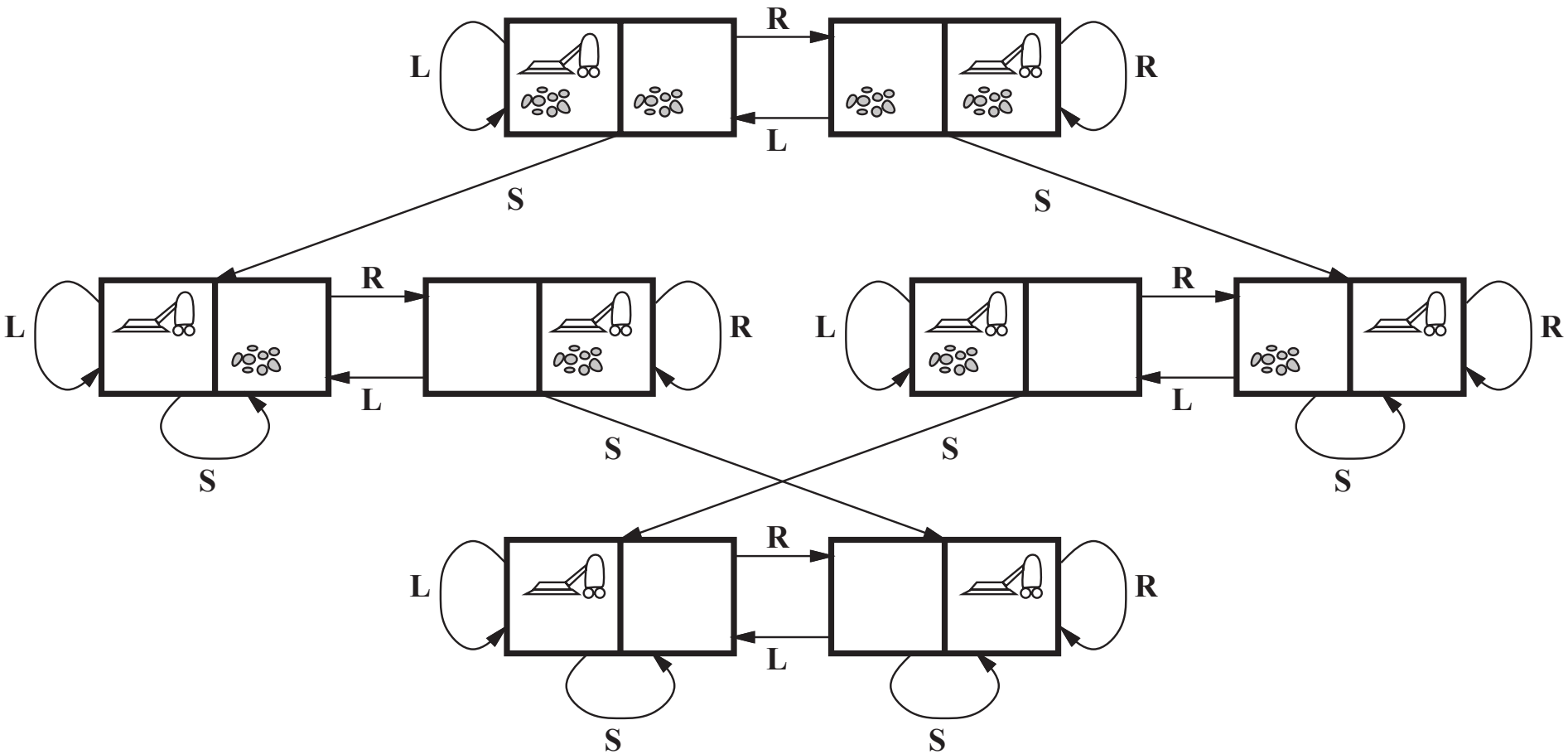
- Stochastic environment
- Multi-agent situation
- Partial observable
- Unknown
- Hierarchical

So When Do You Need Planning?

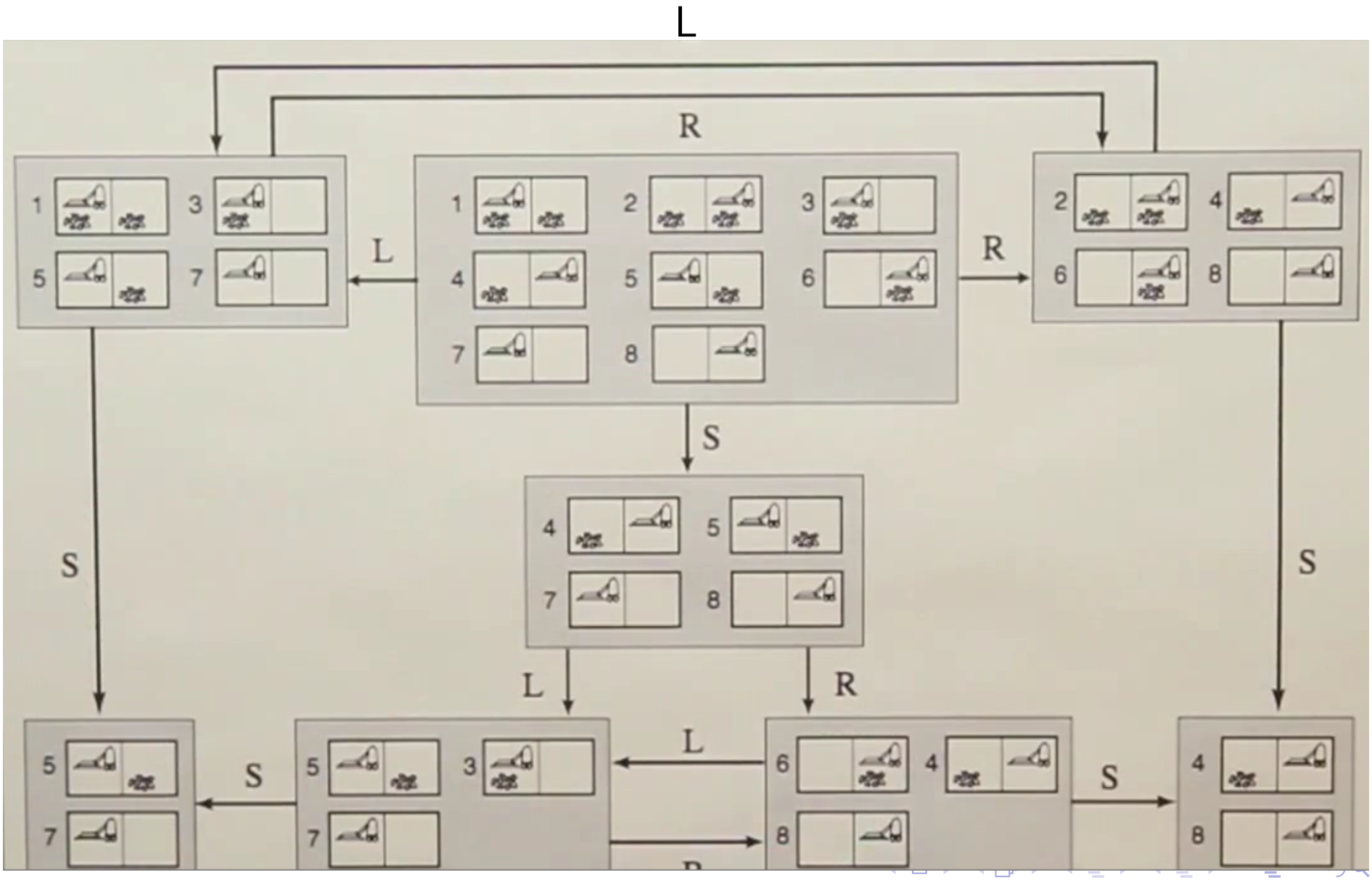
- Stochastic environment
- Multi-agent situation
- Partial observable
- Unknown
- Hierarchical

In these cases, may need to plan in **belief states**

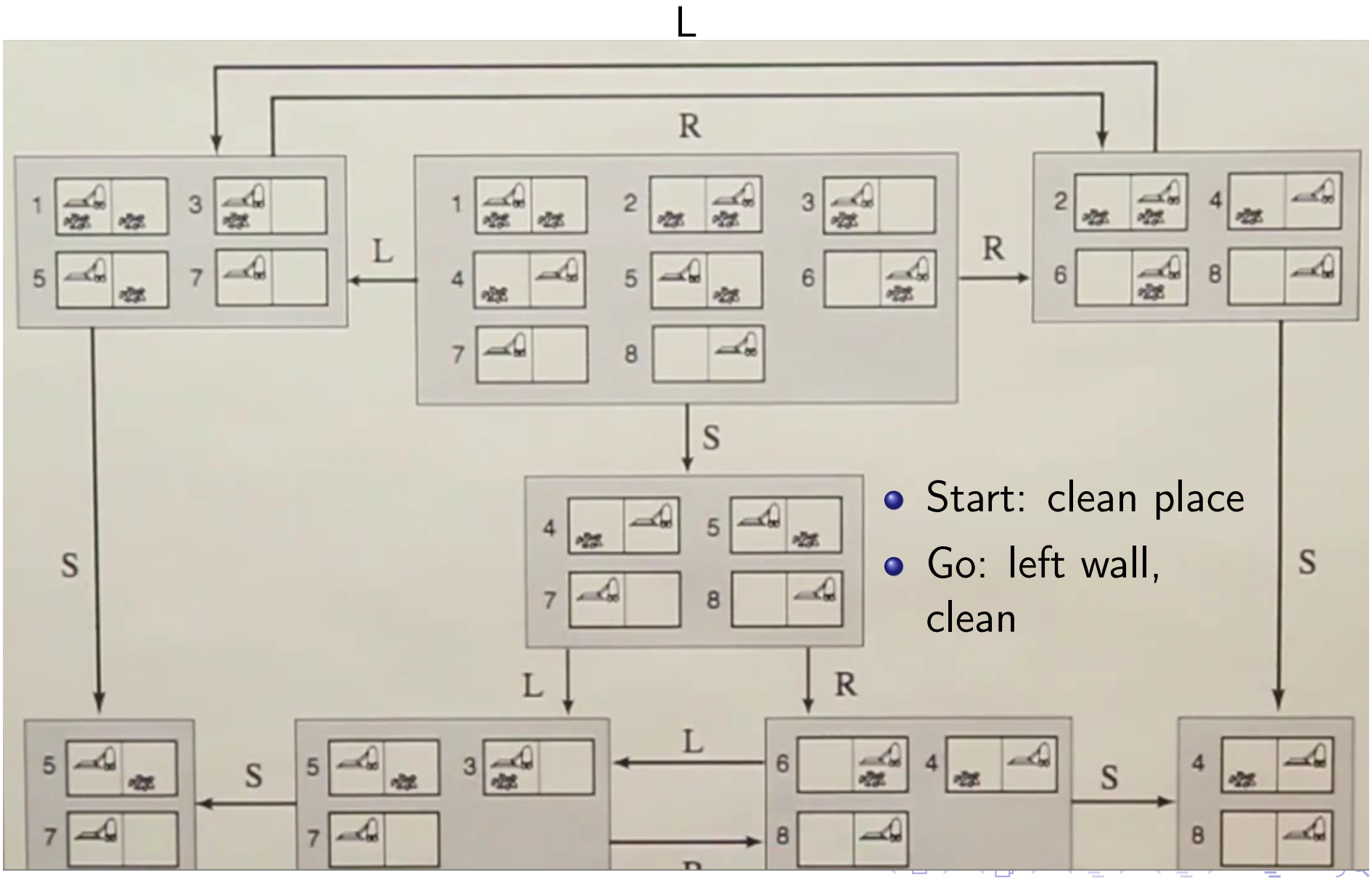
Sensorless World



Sensorless World

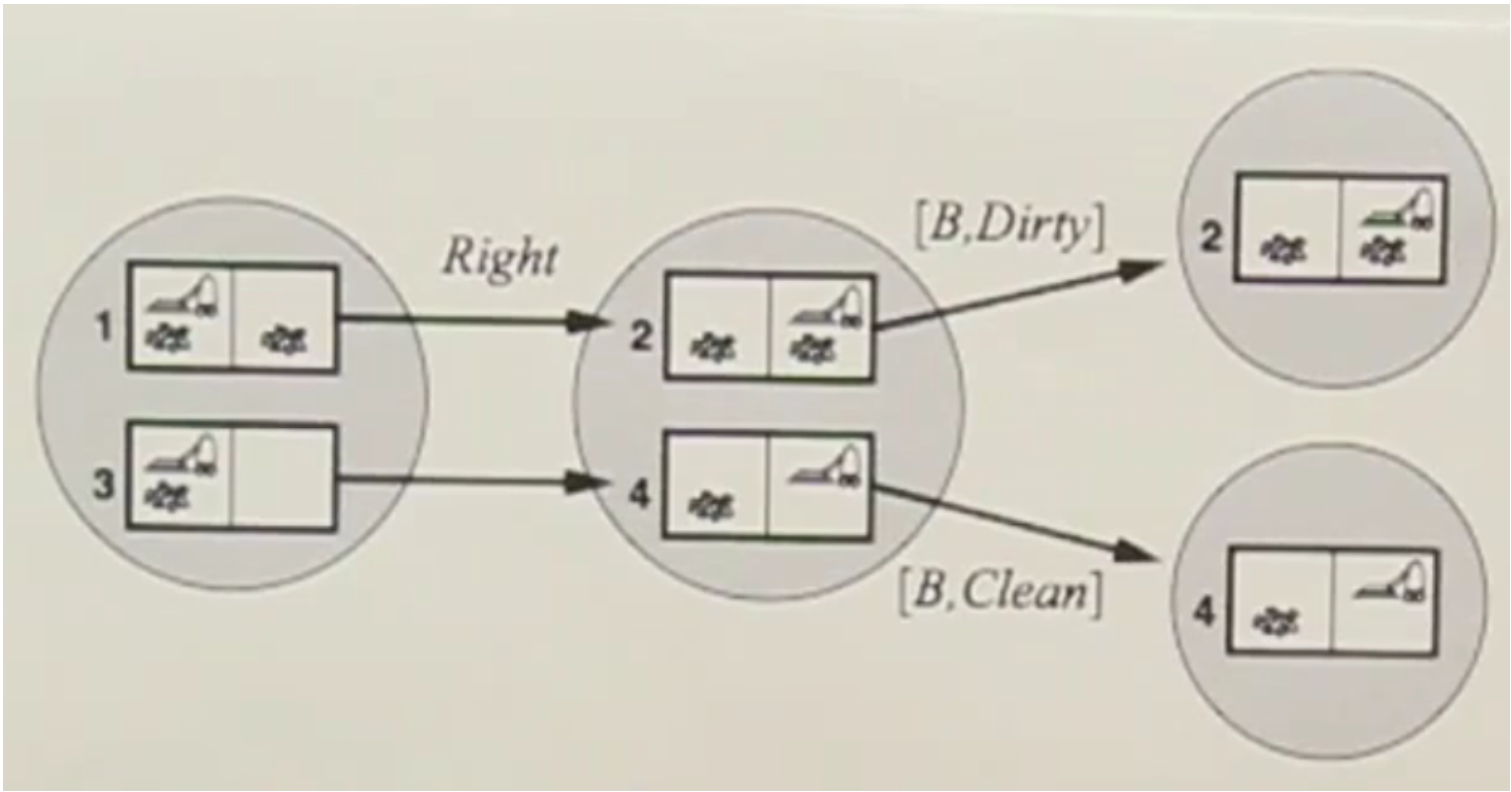


Sensorless World



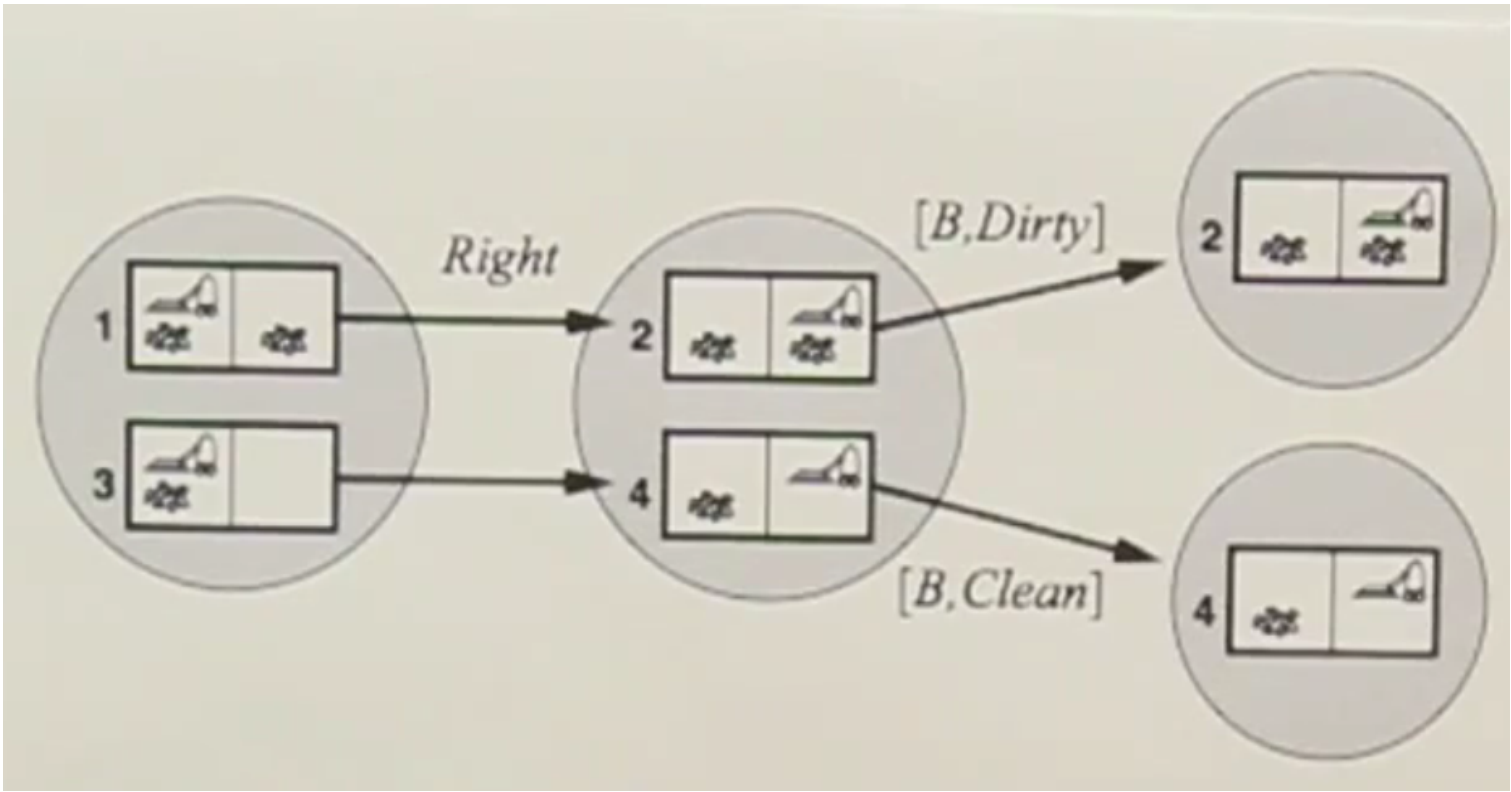
- Start: clean place
- Go: left wall, clean

Partially Observable Environment



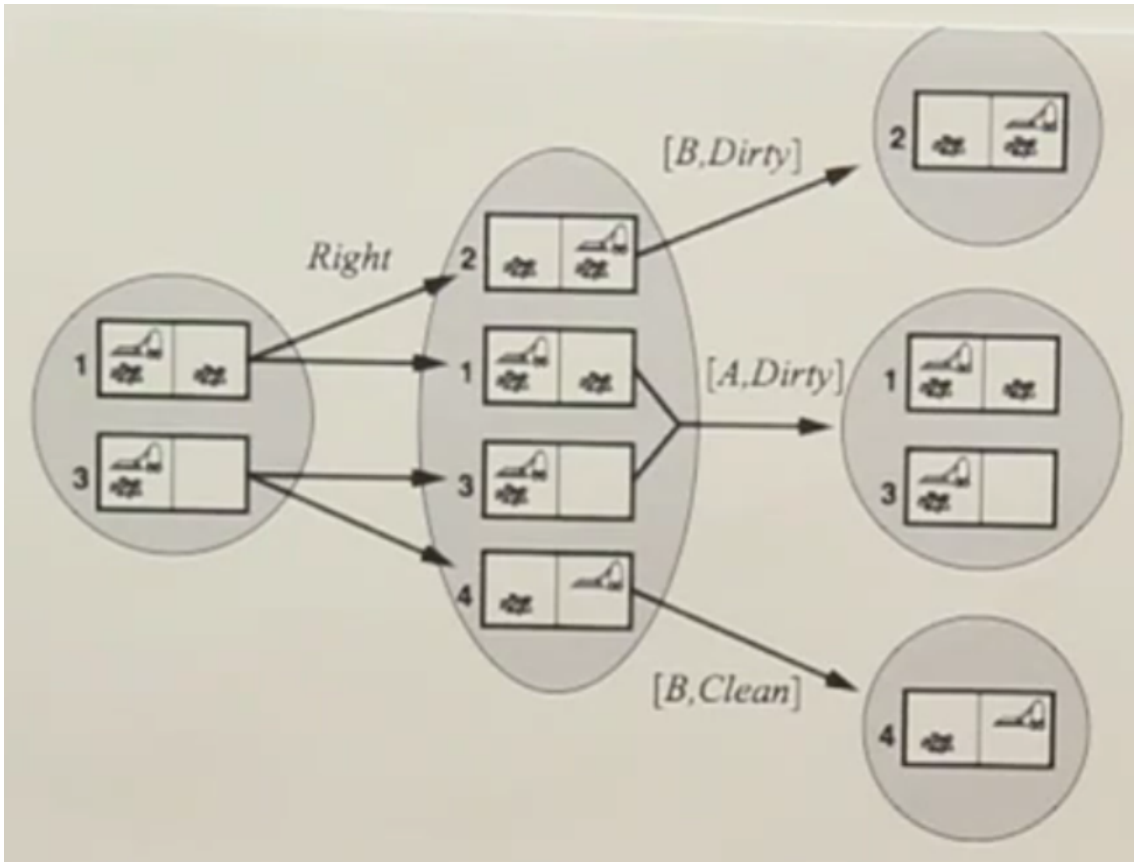
- Deterministic, but sense only local dirt & location

Partially Observable Environment



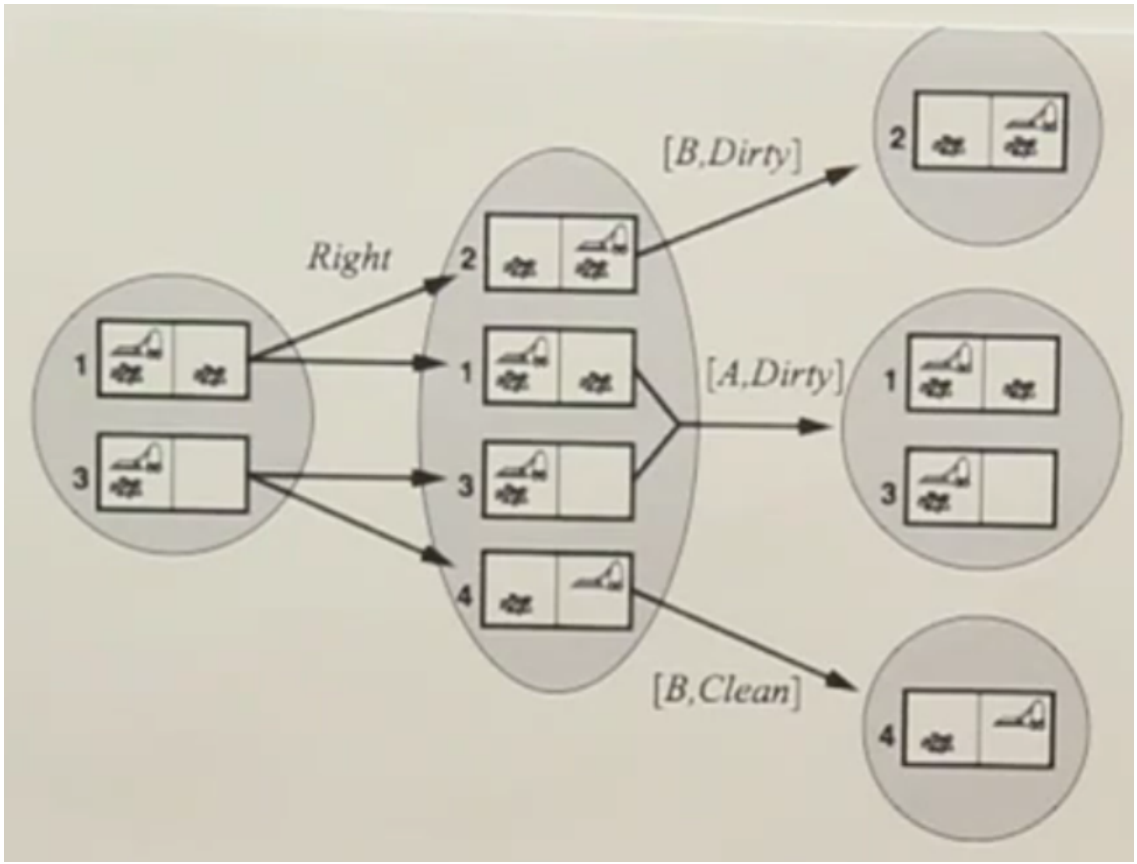
- Deterministic, but sense only local dirt & location
- Belief states get **smaller** with actions

Stochastic (Random) Environment



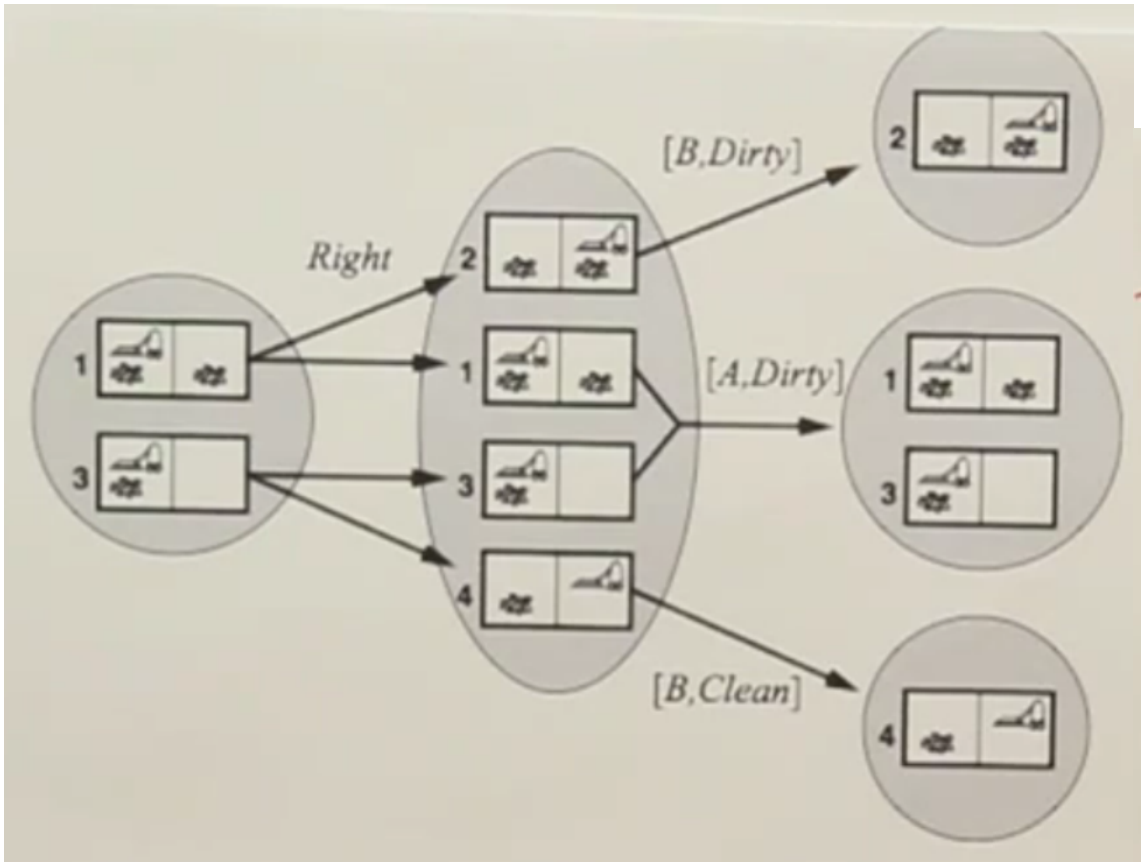
- Still local sensing
- Robot has slippery wheels, may not move

Stochastic (Random) Environment

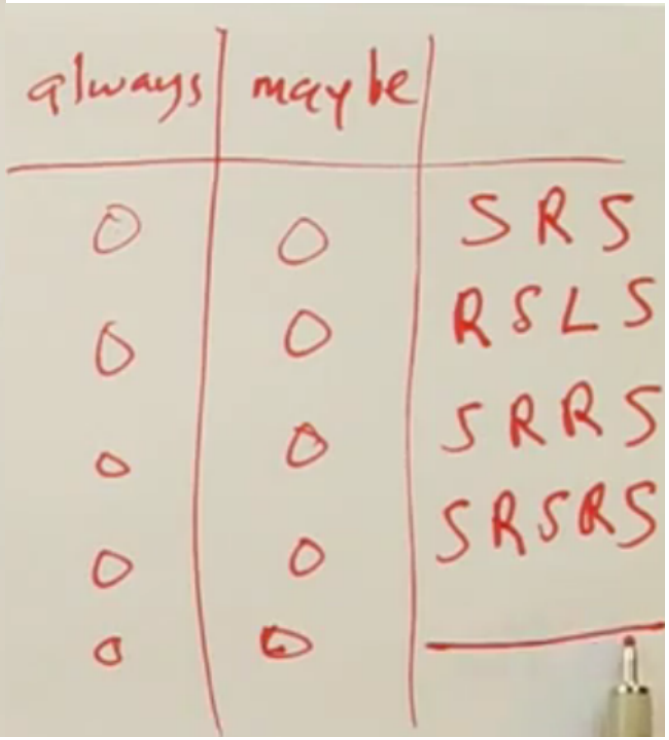


- Still local sensing
- Robot has slippery wheels, may not move
- Belief states get **larger** with actions

Stochastic (Random) Environment

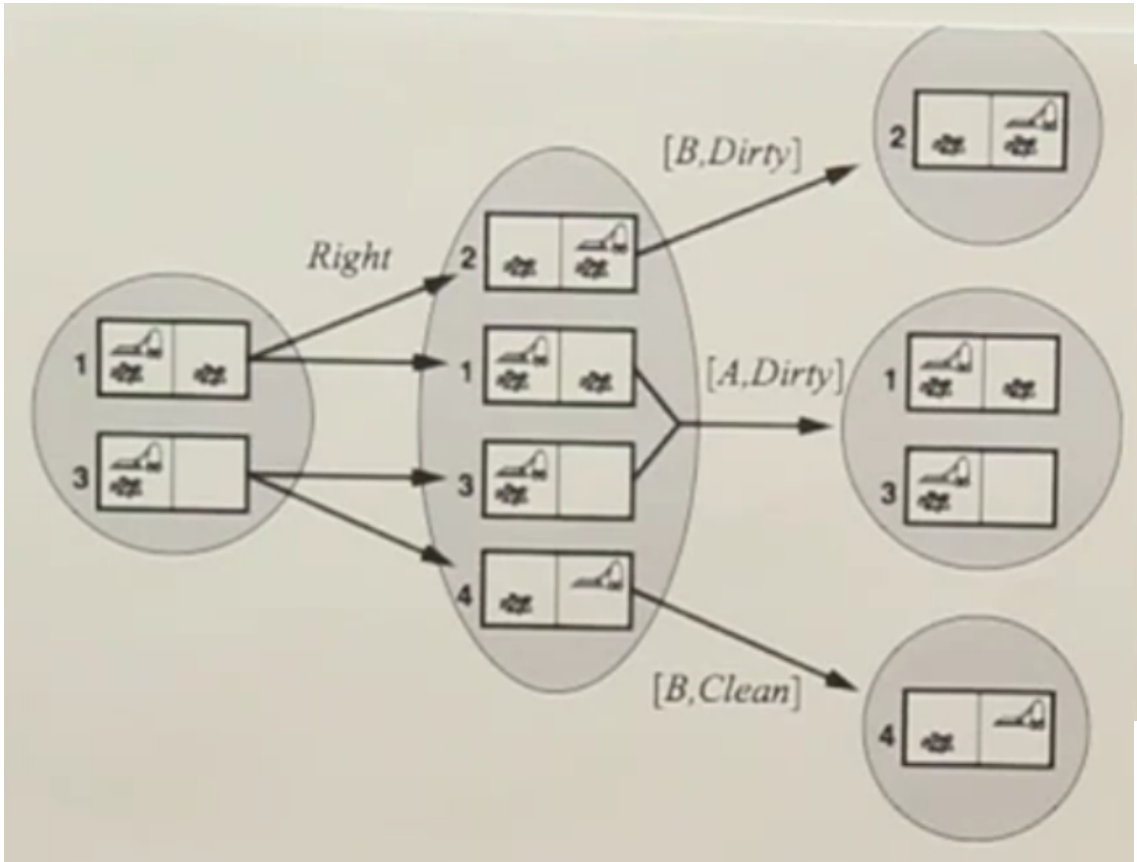


Can reach clean world?

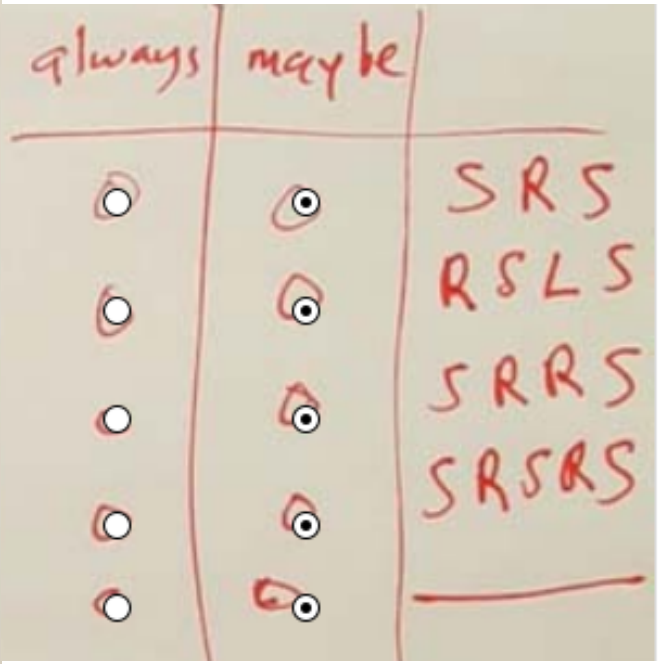


- Still local sensing
- Robot has slippery wheels, may not move
- Belief states get **larger** with actions

Stochastic (Random) Environment



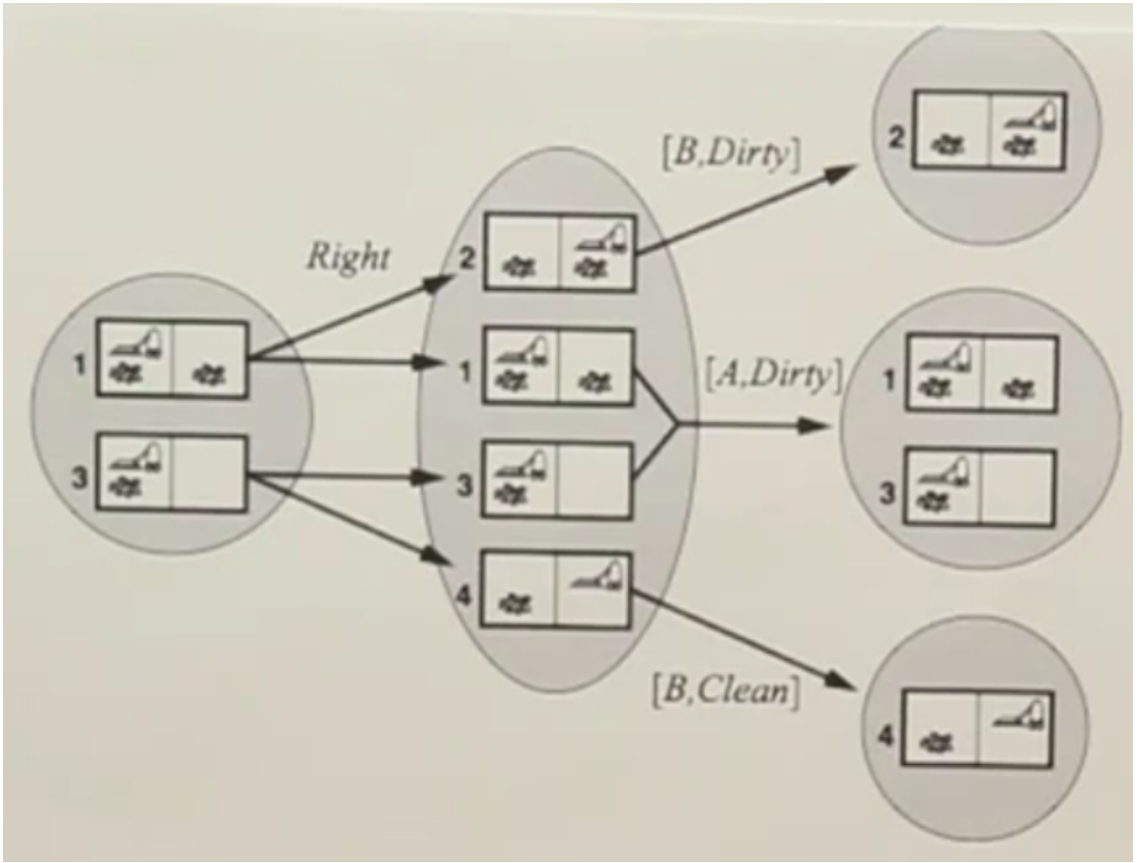
Can reach clean world?



No finite solution!

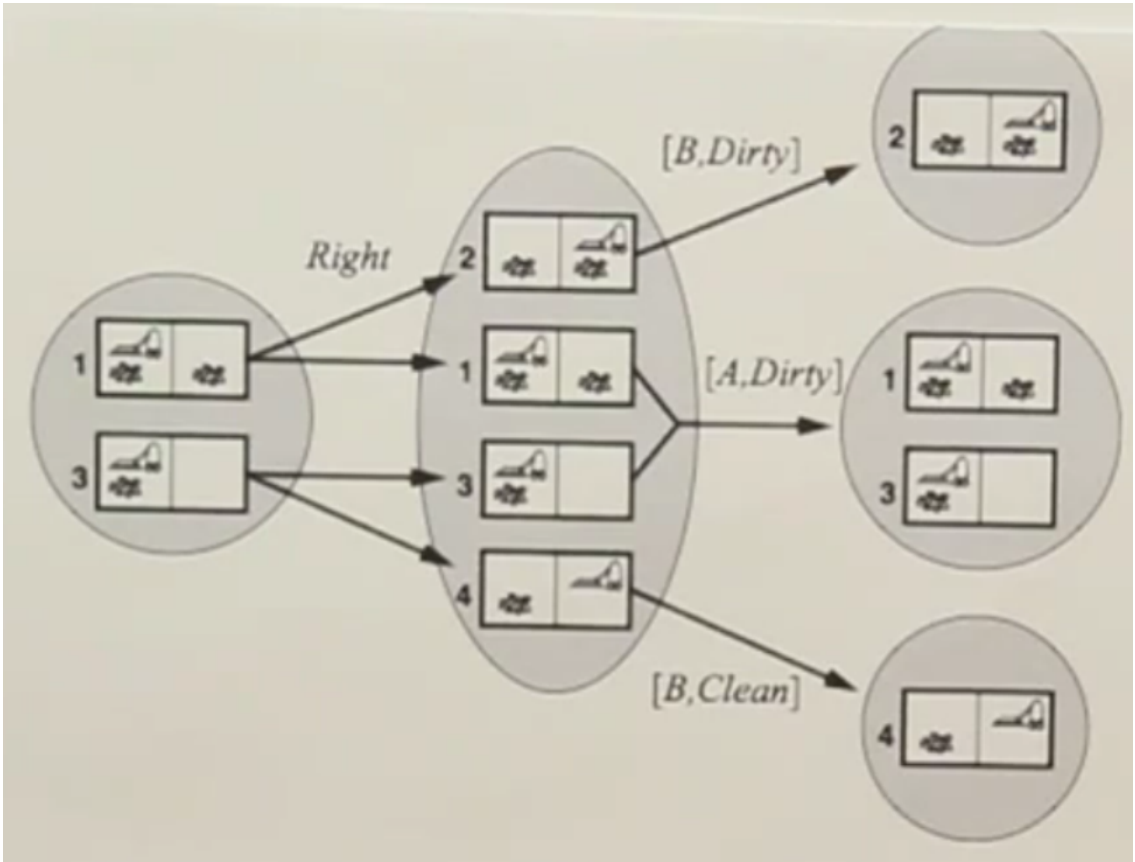
- Still local sensing
- Robot has slippery wheels, may not move
- Belief states get **larger** with actions

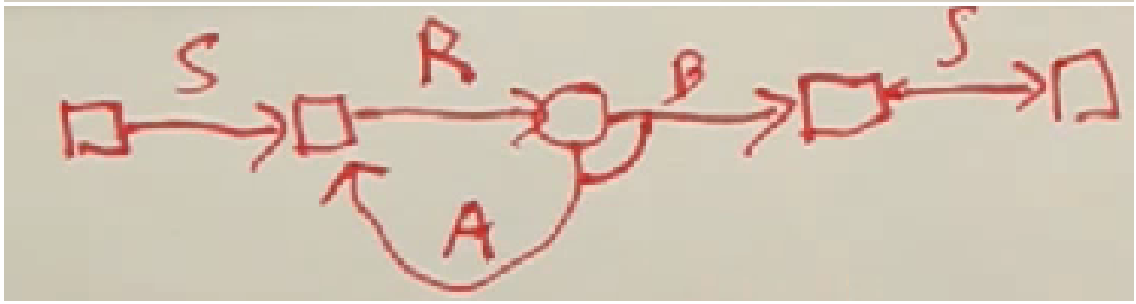
Infinite Sequences?



- Finite sequences don't work:
  - $[R, S, R, S]$

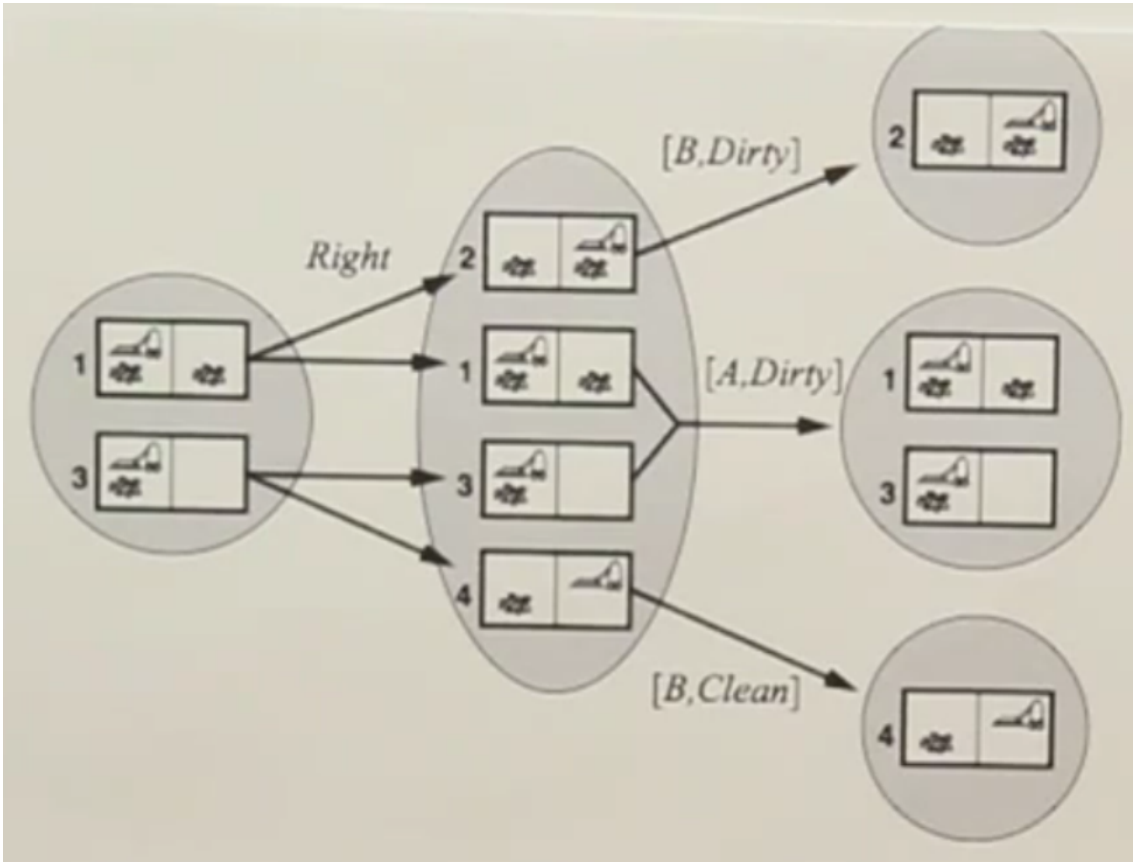
Infinite Sequences?

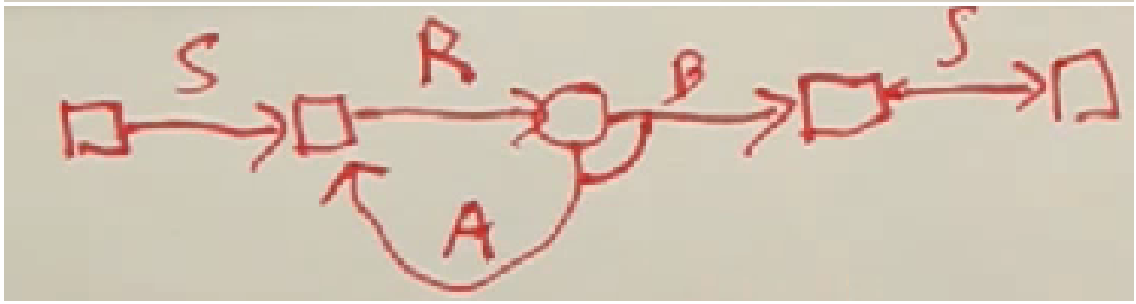




- Finite sequences don't work:
  - $[R, S, R, S]$
- Make a finite plan with branches

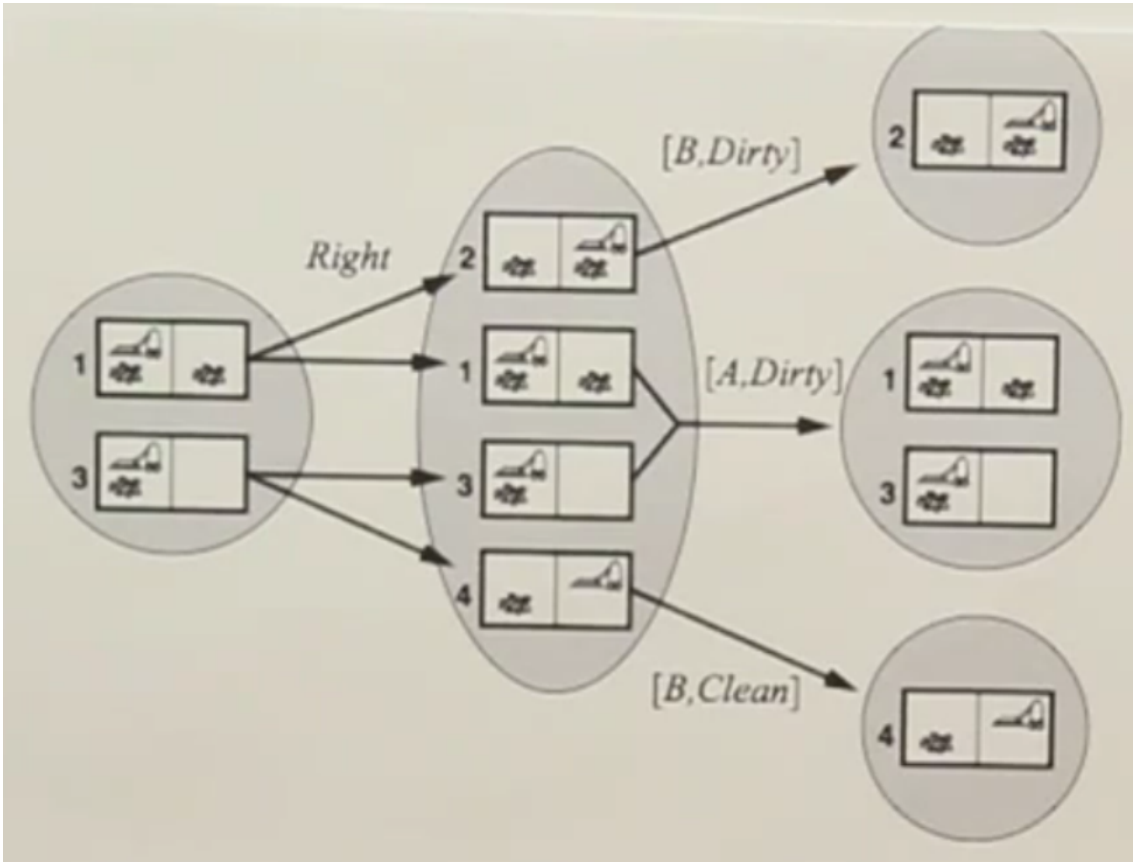
Infinite Sequences?

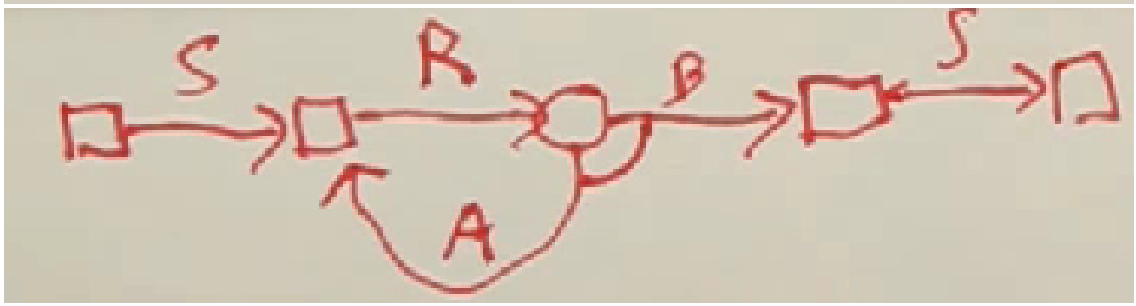




- Finite sequences don't work:
  - $[R, S, R, S]$
- Make a finite plan with branches
- Gives an infinite sequence!

Infinite Sequences?



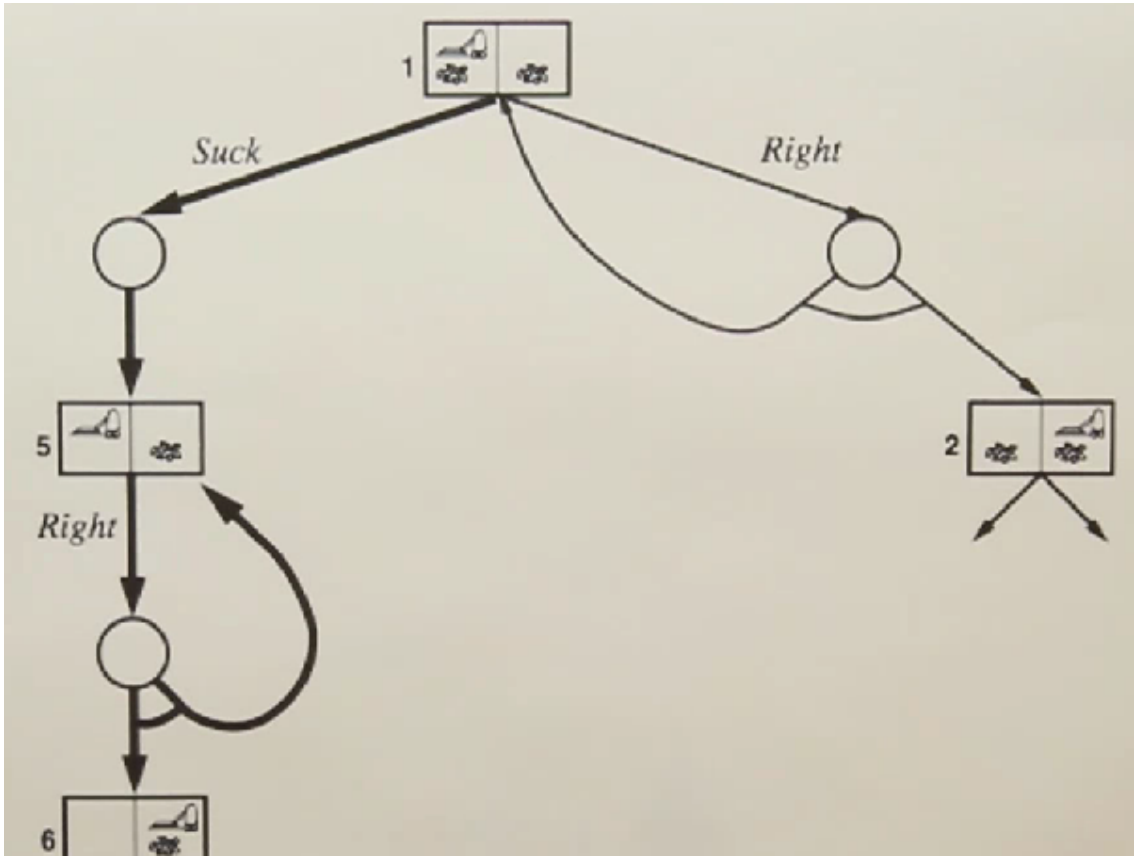


- Finite sequences don't work:
  - $[R, S, R, S]$
- Make a finite plan with branches
- Gives an infinite sequence!

Can write it like:

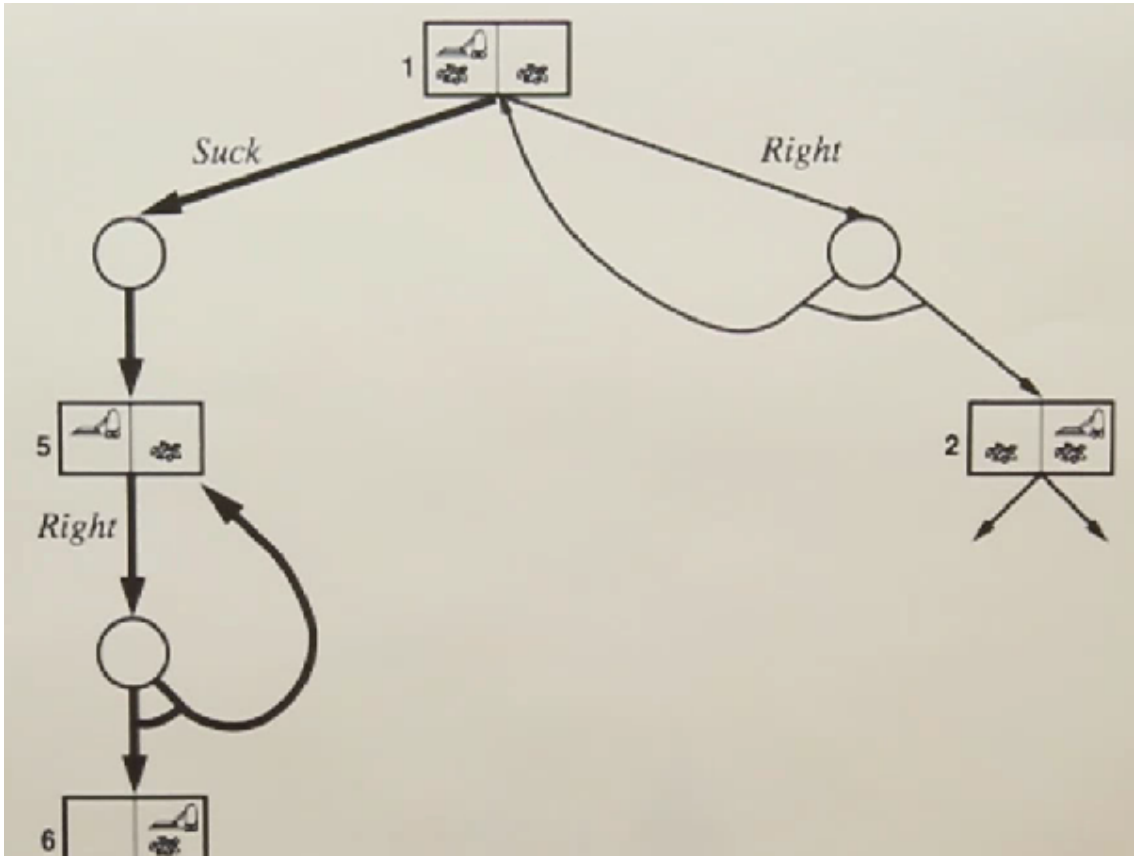
$$[S, \text{While } A: \ R, S]$$

Searching For a Plan



Like searching for paths, find a plan that reaches the goal.

Searching For a Plan



Unbounded solution:

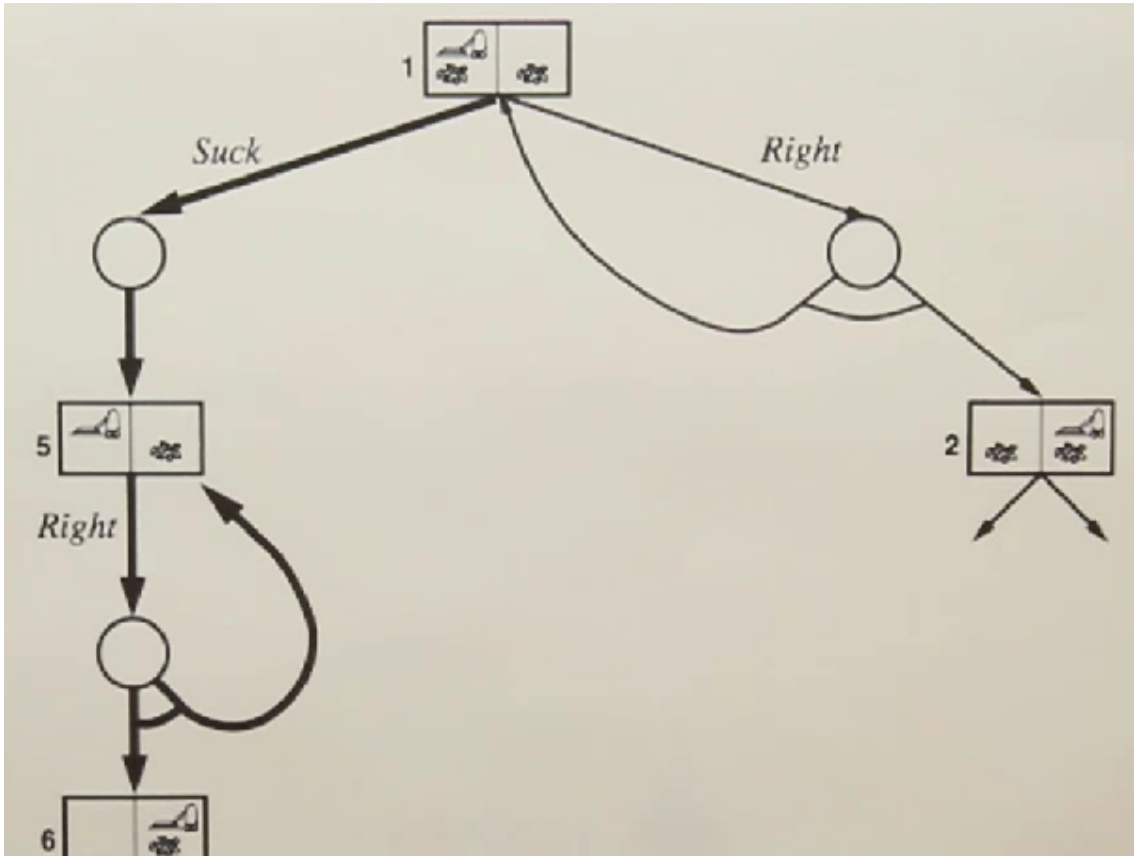
- Some leaf goal
- Every leaf goal
- No

Bounded solution:

- No branch
- No loops
- No

Like searching for paths, find a plan that reaches the goal.

Searching For a Plan



Unbounded solution:

- Some leaf goal
- **Every leaf goal**
- No

Bounded solution:

- No branch
- **No loops**
- No

Like searching for paths, find a plan that reaches the goal.

Mathematical Notation

$$[A, S, F] \quad \text{Result}(\text{Result}(A, A \to S), S \to F) \in \text{Goals}$$

Deterministic:

$$s' = \text{Result}(a, s)$$

where s is state and a is action.

Mathematical Notation

$$[A, S, F] \quad \text{Result}(\text{Result}(A, A \to S), S \to F) \in \text{Goals}$$

Deterministic:

$$s' = \text{Result}(a, s)$$

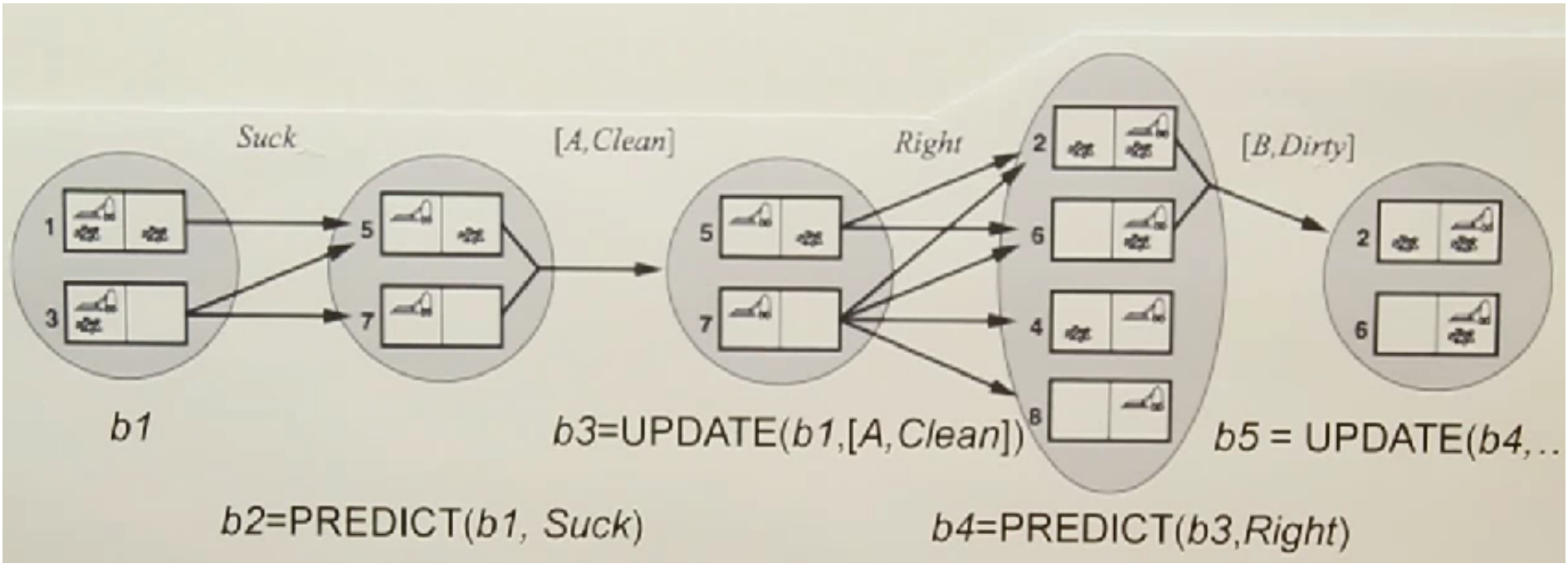
where s is state and a is action.

Non-deterministic: Predict-update beliefs (b) cycle:

$$b' = \text{Update}(\text{Predict}(b, a), o)$$

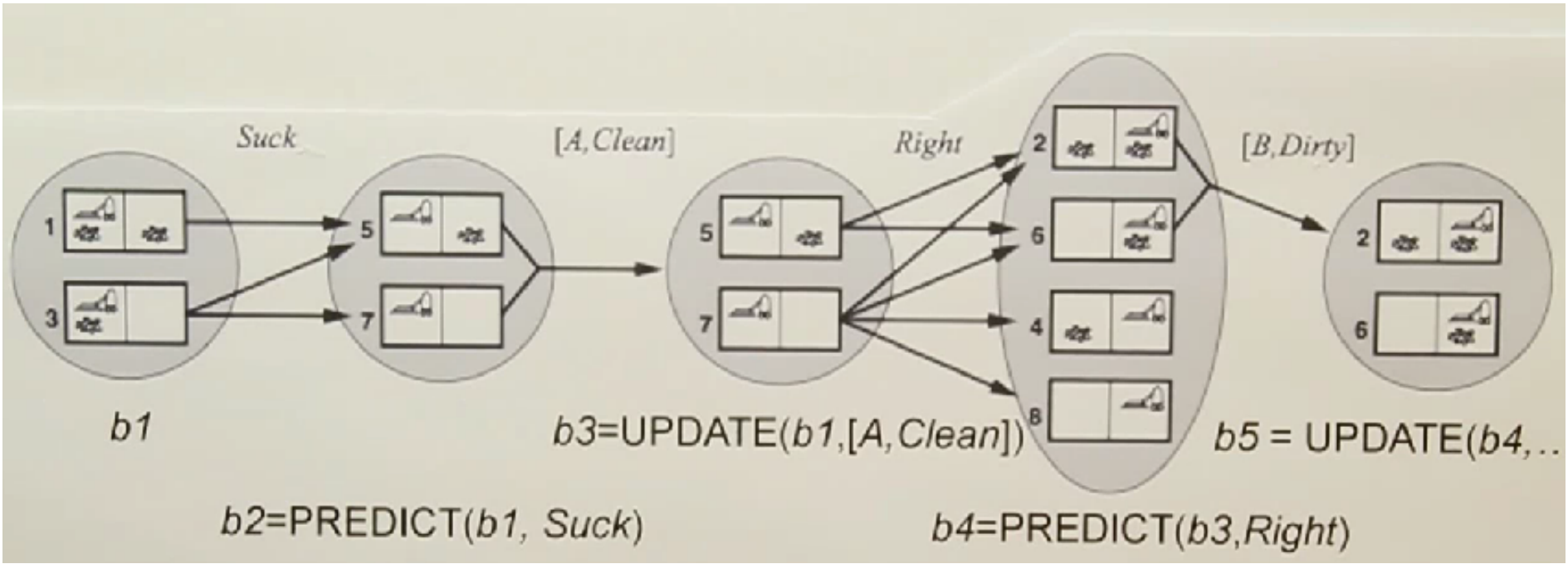
where o is observation.

Predict-Update Cycle



- Babies randomly dirtying floors

Predict-Update Cycle



- Babies randomly dirtying floors

Problems:

- Large solution, simpler to list world state as variables

Classical Planning

State space: k boolean variables; size:

Classical Planning

State space: k boolean variables; size: 2^k

Classical Planning

State space: k boolean variables; size: 2^k

World state: Complete assignment

Belief state: Complete/partial/arbitrary formula

- Called an **action schema**

Action Schemas

1. Action
2. Precondition
3. Effect

Action Schemas

1. Action
2. Precondition
3. Effect

Example:
Action(Fly(p, $from$, to),
  Pre: $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
  Eff: $\neg At(p, from) \wedge At(p, to)$)

Action Schemas

1. Action
2. Precondition
3. Effect

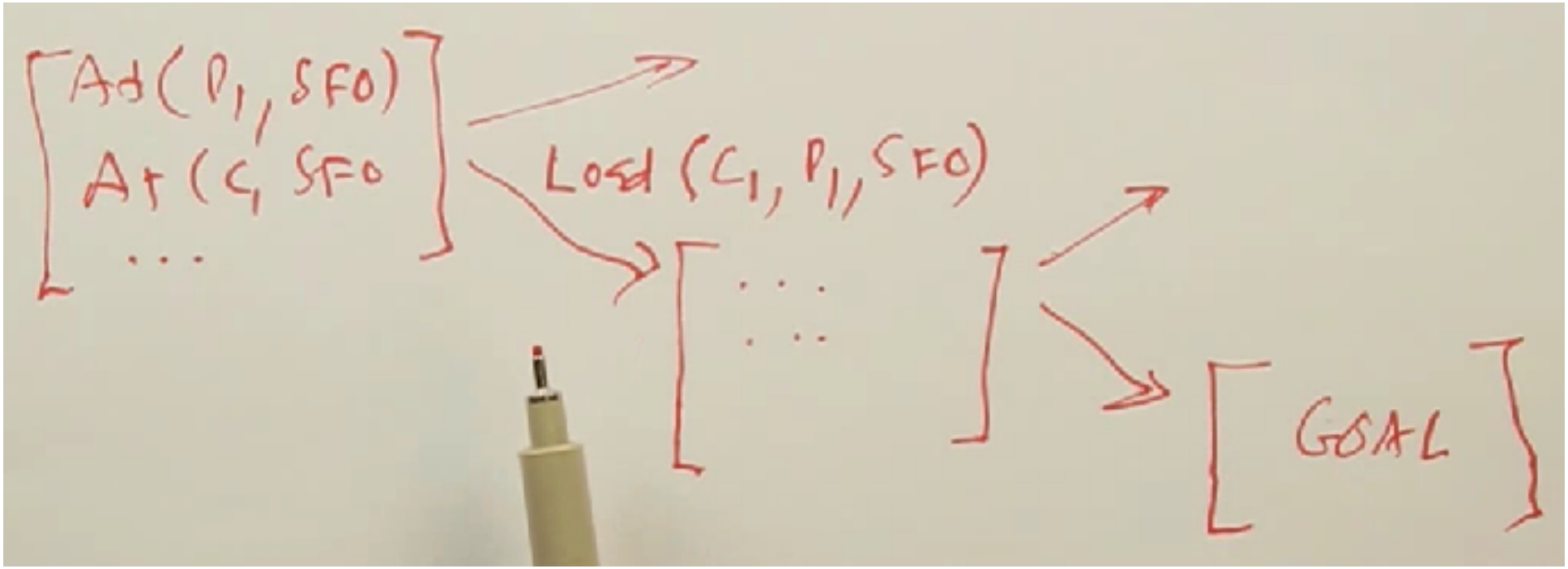
Example:
Action(Fly(p, $from$, to),
  Pre: $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
  Eff: $\neg At(p, from) \wedge At(p, to)$)

Is it FOL?

Air Cargo Example

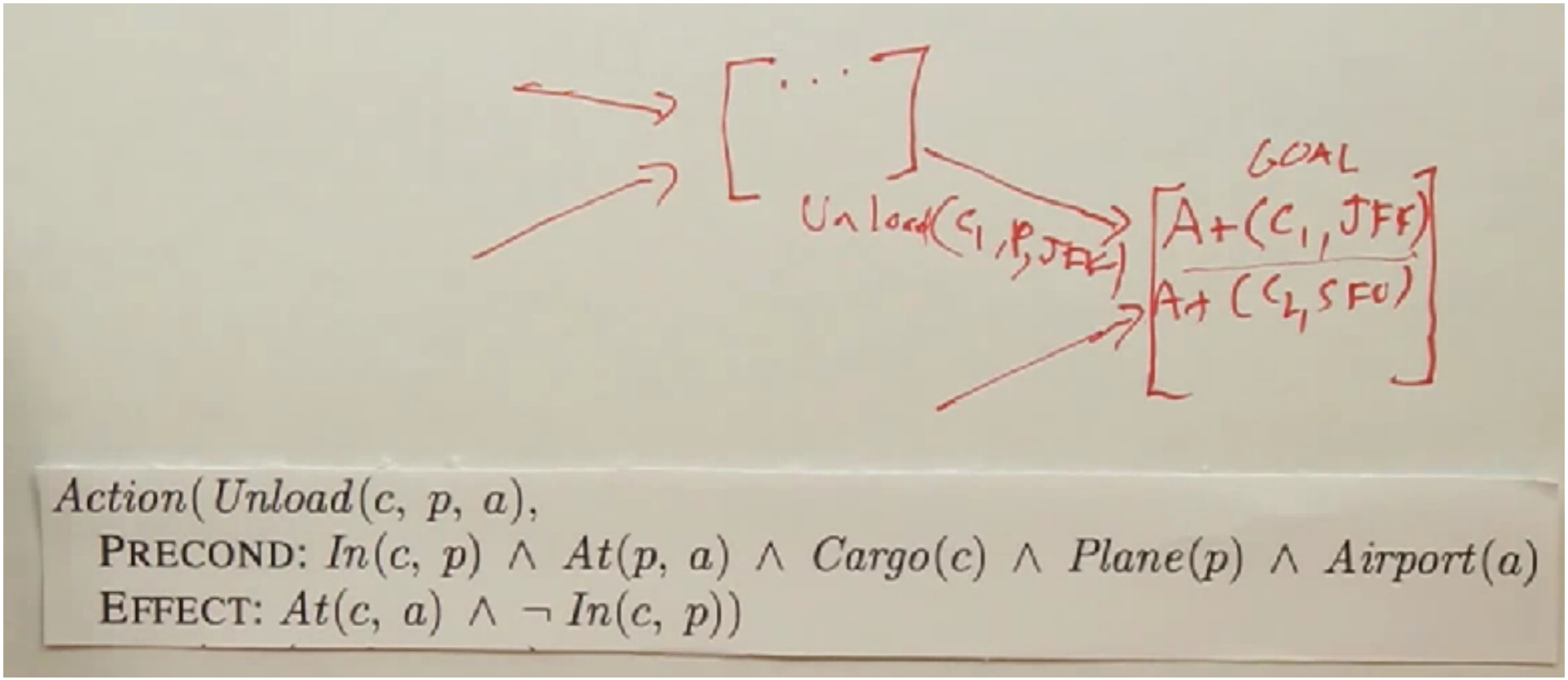
$$
\begin{aligned}
&Init(At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land At(P_2, JFK) \\
&\quad \land Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2) \\
&\quad \land Airport(JFK) \land Airport(SFO)) \\
&Goal(At(C_1, JFK) \land At(C_2, SFO)) \\
&Action(Load(c, p, a), \\
&\quad \text{PRECOND: } At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a) \\
&\quad \text{EFFECT: } \neg At(c, a) \land In(c, p)) \\
&Action(Unload(c, p, a), \\
&\quad \text{PRECOND: } In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a) \\
&\quad \text{EFFECT: } At(c, a) \land \neg In(c, p)) \\
&Action(Fly(p, from, to), \\
&\quad \text{PRECOND: } At(p, from) \land Plane(p) \land Airport(from) \land Airport(to) \\
&\quad \text{EFFECT: } \neg At(p, from) \land At(p, to))
\end{aligned}
$$

Progression/Forward Search



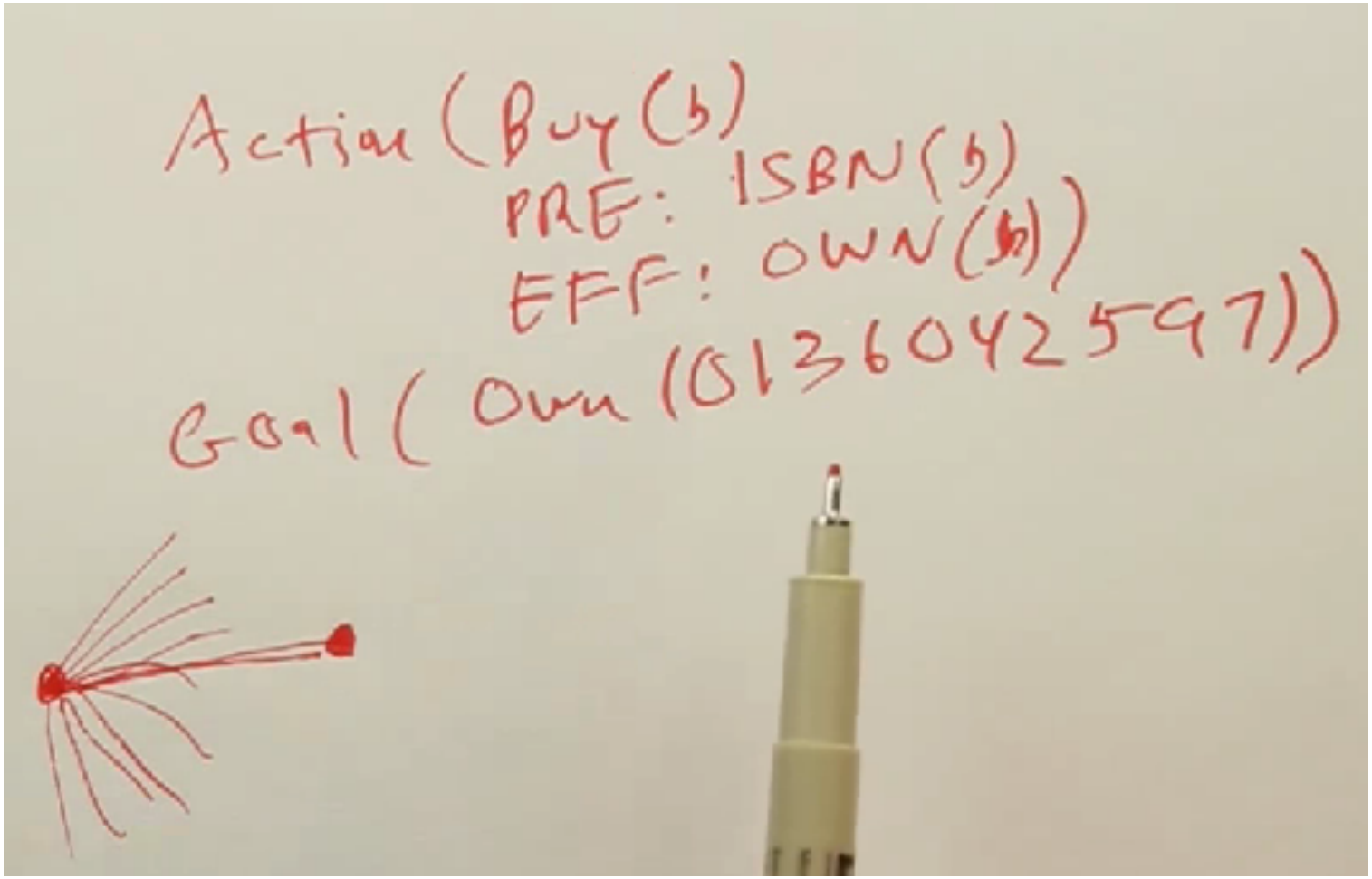
- Like regular search.

Regression/Backward Search



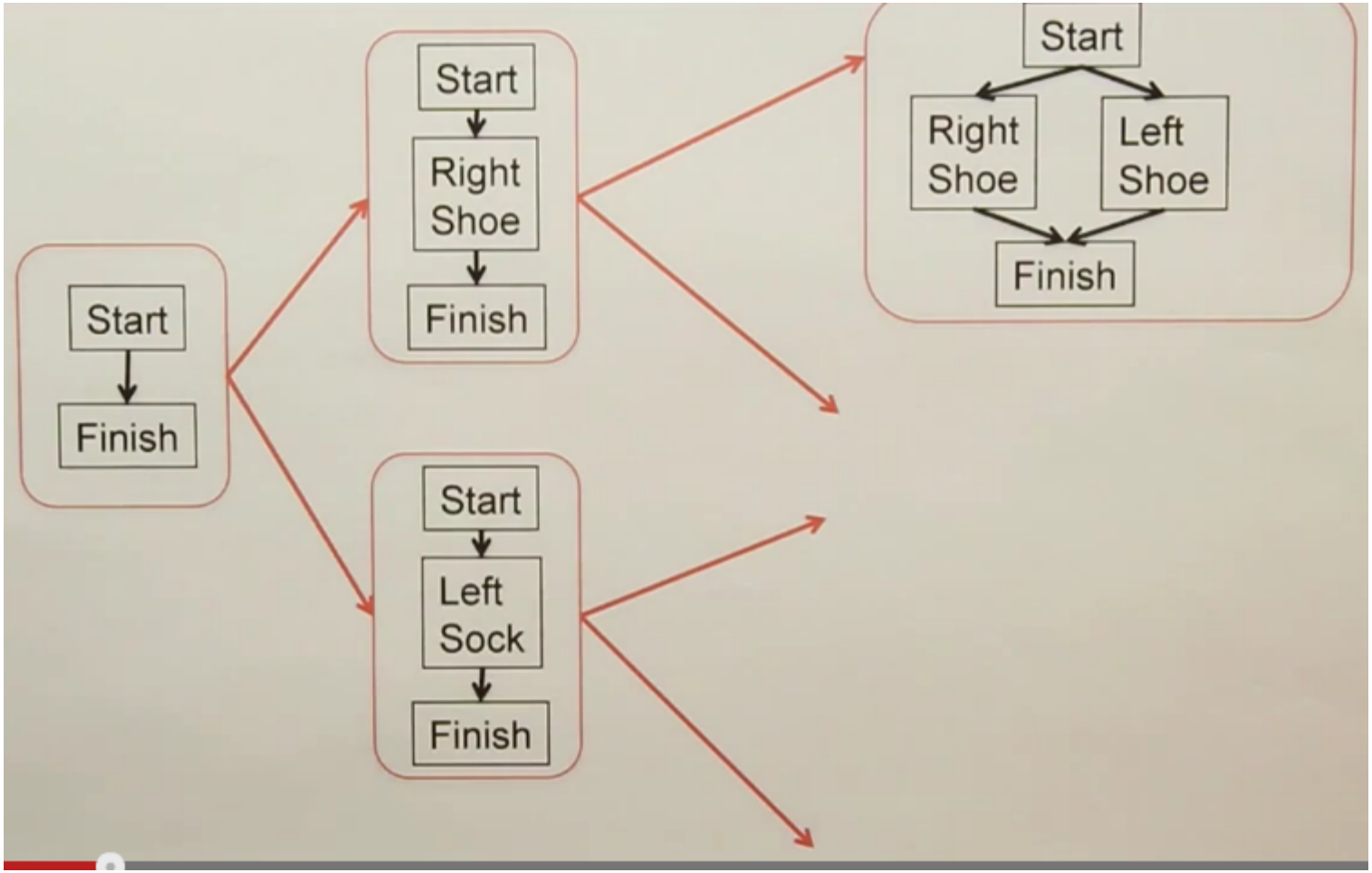
- Start with goal, have many unknowns.

When to Choose Forward vs. Backward?

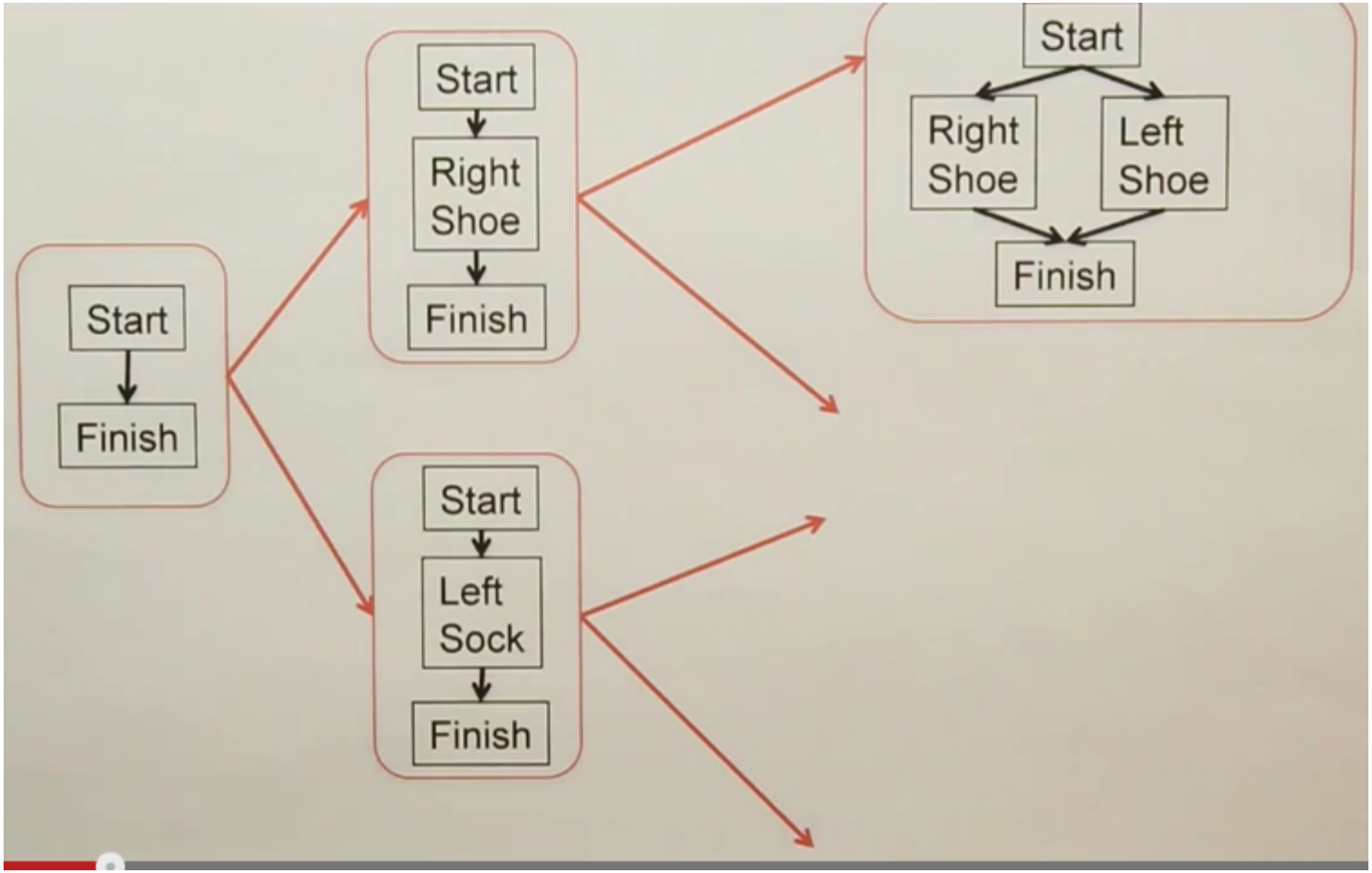


- Remember when to choose breadth-first vs. depth first.
- Fewer options when starting from the goal.
- Can think of other examples?

Searching the "Plan Space"



Searching the "Plan Space"



- How would one use genetic algorithms?

Current Trend: Forward Search with Heuristics

Tiles example:

Action(Slide(t, a, b),
  Pre: $\mathrm{On}(t, a) \wedge \mathrm{Tile}(t) \wedge \mathrm{Blank}(b) \wedge \mathrm{Adj}(a, b)$
  Eff: $\mathrm{On}(t, b) \wedge \mathrm{Blank}(a) \wedge \neg\mathrm{On}(t, a) \wedge \neg\mathrm{Blank}(b)$)

Current Trend: Forward Search with Heuristics

Tiles example:

Action(Slide(t, a, b),
  Pre: $\mathrm{On}(t, a) \wedge \mathrm{Tile}(t) \wedge \mathrm{Blank}(b) \wedge \mathrm{Adj}(a, b)$
  Eff: $\mathrm{On}(t, b) \wedge \mathrm{Blank}(a) \wedge \neg\mathrm{On}(t, a) \wedge \neg\mathrm{Blank}(b)$)

How to use heuristics? Remember Romania routes.

Current Trend: Forward Search with Heuristics

Tiles example:

Action(Slide(t, a, b),
  Pre: $\mathrm{On}(t, a) \wedge \mathrm{Tile}(t) \wedge \mathrm{Blank}(b) \wedge \mathrm{Adj}(a, b)$
  Eff: $\mathrm{On}(t, b) \wedge \mathrm{Blank}(a) \wedge \neg\mathrm{On}(t, a) \wedge \neg\mathrm{Blank}(b)$)

How to use heuristics? Remember Romania routes.

- Choose approximate solutions and use during search
- Delete terms from schema

Current Trend: Forward Search with Heuristics

Tiles example:

Action(Slide(t, a, b),
  Pre: $\mathrm{On}(t, a) \wedge \mathrm{Tile}(t) \wedge \mathrm{Blank}(b) \wedge \mathrm{Adj}(a, b)$
  Eff: $\mathrm{On}(t, b) \wedge \mathrm{Blank}(a) \wedge \neg\mathrm{On}(t, a) \wedge \neg\mathrm{Blank}(b)$)

How to use heuristics? Remember Romania routes.

- Choose approximate solutions and use during search
- Delete terms from schema
- Can also do this programmatically

One Last Bit: Situation Calculus

Uses First Order Logic (FOL) for planning:

- **Actions:** are objects; e.g., $\mathrm{Fly}(p, x, y)$
- **Situations:** objects; e.g., $s' = \mathrm{Result}(s, a)$
- **Fluents:** change at each situation: e.g., $\mathrm{At}(p, x, s)$
- **Possible actions:** objects; $\mathrm{Poss}(a, s)$
  Written as in: $\mathrm{Pre}(s) \Rightarrow \mathrm{Poss}(a, s)$

One Last Bit: Situation Calculus

Uses First Order Logic (FOL) for planning:

- **Actions:** are objects; e.g., $\mathrm{Fly}(p, x, y)$
- **Situations:** objects; e.g., $s' = \mathrm{Result}(s, a)$
- **Fluents:** change at each situation: e.g., $\mathrm{At}(p, x, s)$
- **Possible actions:** objects; $\mathrm{Poss}(a, s)$
  Written as in: $\mathrm{Pre}(s) \Rightarrow \mathrm{Poss}(a, s)$

Plane example:

$$\mathrm{Plane}(p, s) \wedge \mathrm{Airport}(x, s) \wedge \mathrm{Airport}(y, s) \wedge \mathrm{At}(p, x, s) \Rightarrow \mathrm{Poss}(\mathrm{Fly}(p, x, y), s)$$

Situation Calculus: Effects

Effects: Successor state axioms:

$$\forall a, s \, \mathrm{Poss}(a, s) \Rightarrow (\text{fluent} \Leftrightarrow \text{caused it} \wedge \text{didn}'\text{t undo it})$$

Situation Calculus: Effects

Effects: Successor state axioms:

$$\forall a, s \, \text{Poss}(a, s) \Rightarrow (\text{fluent} \Leftrightarrow \text{caused it} \wedge \text{didn}'\text{t undo it})$$

Cargo example:

$$\text{Poss}(a, s) \Rightarrow \text{In}(c, p, \text{Result}(s, a)) \quad \Leftrightarrow$$
$$(a = \text{Load}(c, p, x) \vee (\text{In}(c, p, s) \wedge a \neq \text{Unload}(c, p, x)))$$

Situation Calculus: Effects

Effects: Successor state axioms:

$$\forall a, s \, \text{Poss}(a, s) \Rightarrow (\text{fluent} \Leftrightarrow \text{caused it} \wedge \text{didn}'\text{t undo it})$$

Cargo example:

$$\text{Poss}(a, s) \Rightarrow \text{In}(c, p, \text{Result}(s, a)) \quad \Leftrightarrow$$
$$(a = \text{Load}(c, p, x) \vee (\text{In}(c, p, s) \wedge a \neq \text{Unload}(c, p, x)))$$

- Can use power of FOL
- Can use existing theorem provers
- Problem: slow