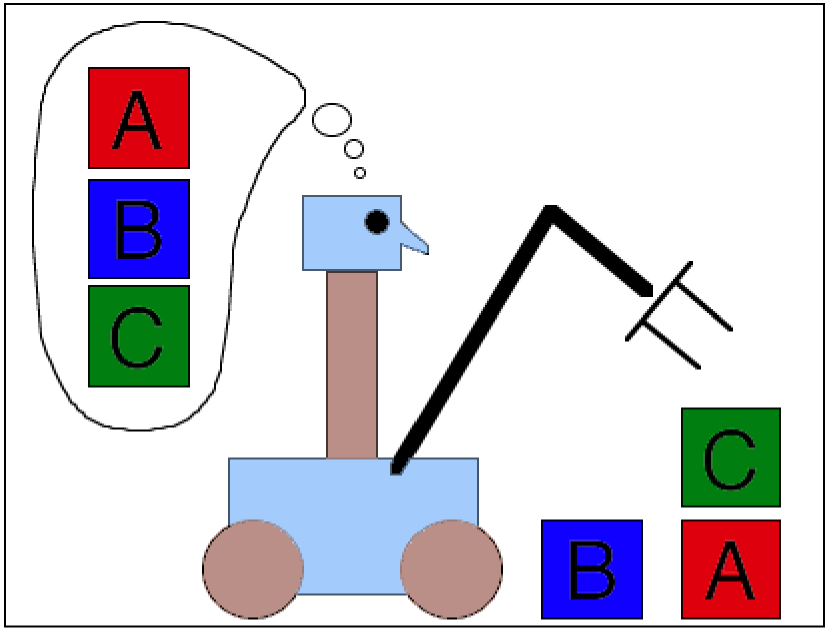




COMP 141: Probabilistic Robotics for Human-Robot Interaction

Instructor: Jivko Sinapov
www.cs.tufts.edu/~jsinapov

This week: Planning



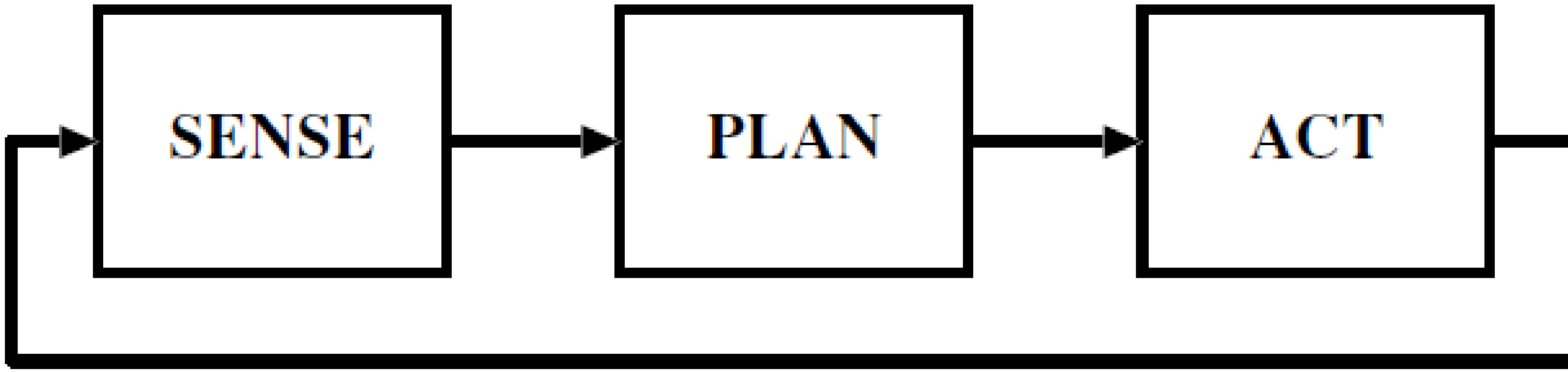
Announcements

HRI 2023

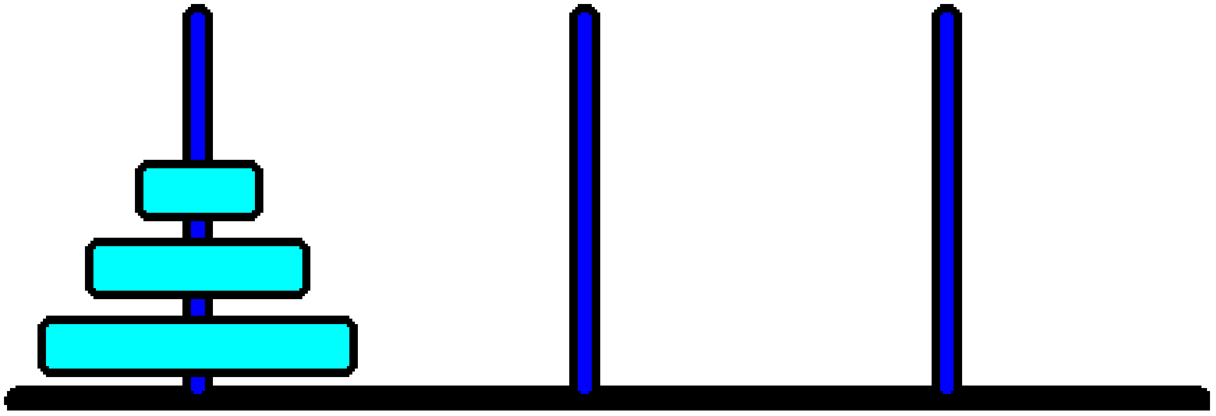
HRI 2023

March 13-16, 2023
HRI 2023 STOCKHOLM, SE
Stockholm, Sweden
Best Video Award
Presented to
Andre Cleaver & Jivko Sinapov
For their contribution
Demonstrating TRAinAR: an Augmented Reality Tool that Helps Humans Teach Robots
Tariq Iqbal Fumihide Tanaka
SIGCHI
IEEE
General chairs
Ginevra Castellano & Laurel Riek
Video / Demo chairs
Tariq Iqbal & Fumihide Tanaka

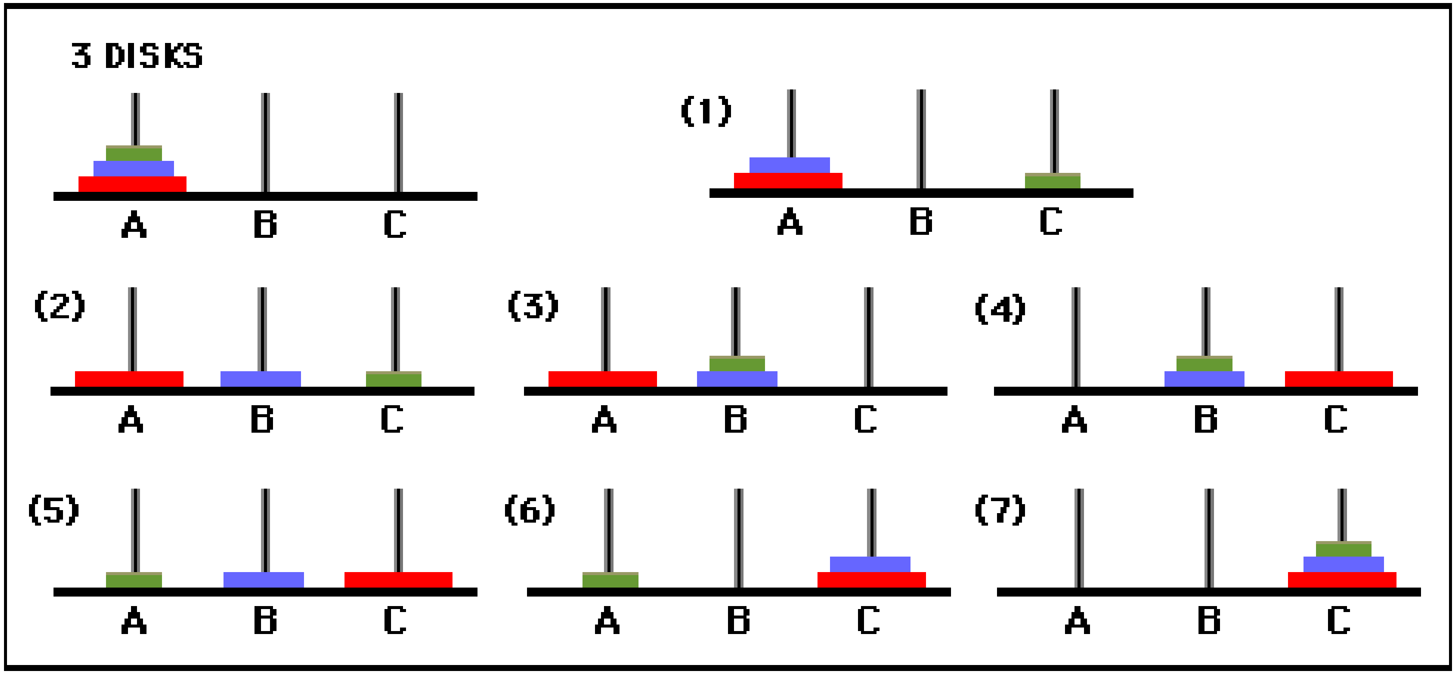
The Early Answer (1967): Sense-Plan-Act



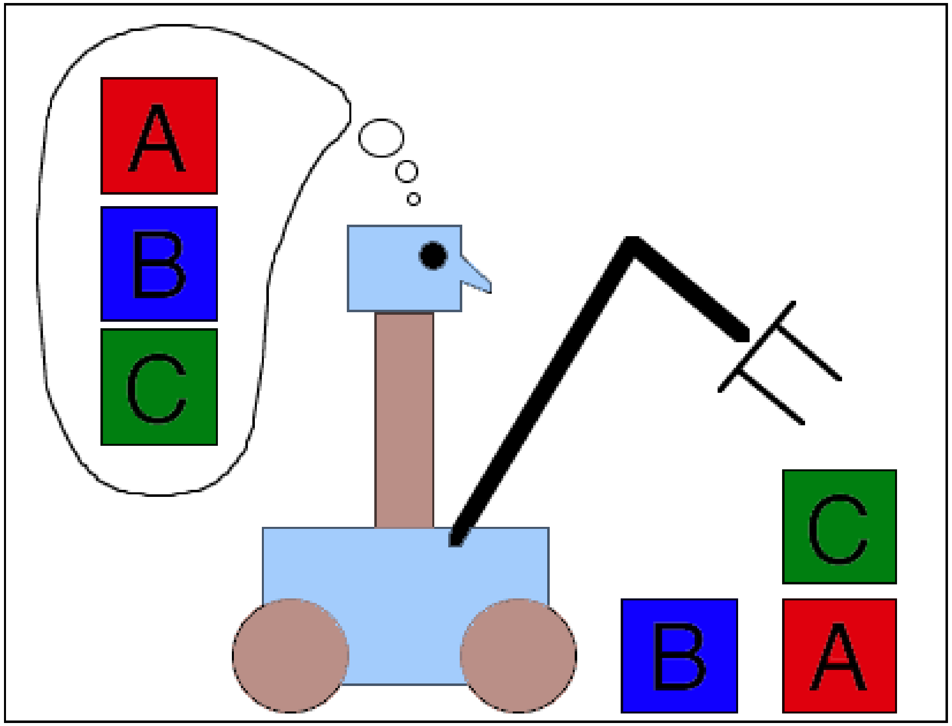
3-Disk Hanoi



Final Plan



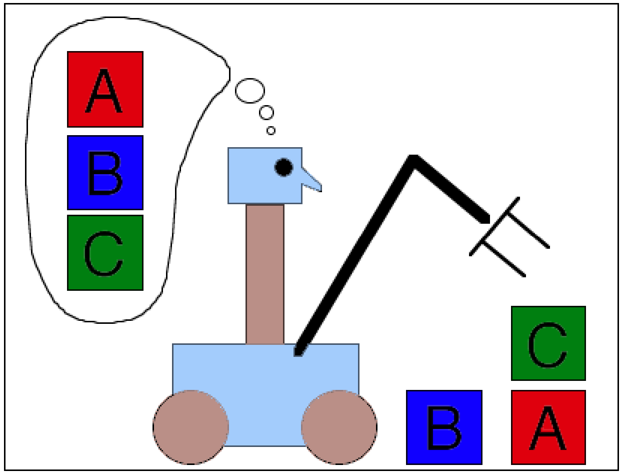
Block-stacking



Block-stacking domain and problem

```
(define (domain blocksworld)
(:requirements :strips :equality)
(:predicates (clear ?x)
             (on-table ?x)
             (arm-empty)
             (holding ?x)
             (on ?x ?y))

(:action pickup
  :parameters (?ob)
  :precondition (and (clear ?ob) (on-table ?ob) (arm-empty))
  :effect (and (holding ?ob) (not (clear ?ob)) (not (on-table ?ob))
               (not (arm-empty))))

(:action putdown
  :parameters  (?ob)
  :precondition (and (holding ?ob))
  :effect (and (clear ?ob) (arm-empty) (on-table ?ob)
               (not (holding ?ob))))

(:action stack
  :parameters  (?ob ?underob)
  //:precondition (and  (clear ?underob) (holding ?ob) (not (= ?ob ?underob)) )
  :precondition (and  (clear ?underob) (holding ?ob))
  :effect (and (arm-empty) (clear ?ob) (on ?ob ?underob)
               (not (clear ?underob)) (not (holding ?ob))))

(:action unstack
  :parameters  (?ob ?underob)
  :precondition (and (on ?ob ?underob) (clear ?ob) (arm-empty))
  :effect (and (holding ?ob) (clear ?underob)
(not (on ?ob ?underob)) (not (clear ?ob)) (not (arm-empty)))))
```

```
(define (problem pb3)
   (:domain blocksworld)
   (:objects a b c)
   (:init (on-table a) (on-table b)   (on c a)
          (clear b) (clear c) (arm-empty))
(:goal (and (on a b) (on b c))))
```

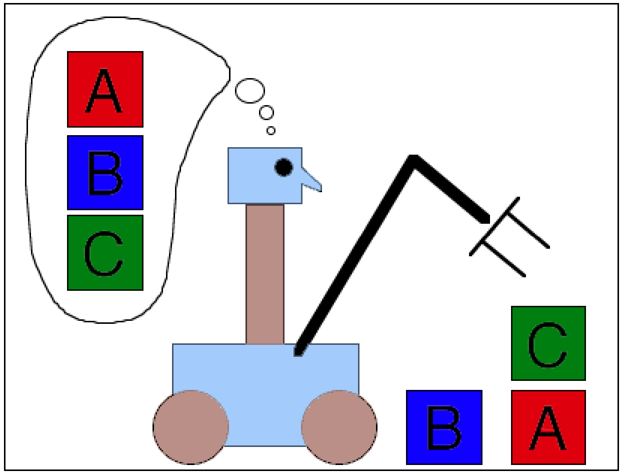


https://github.com/gerryai/PDDL4J/blob/master/pddl/blockworld/blocksworld.pddl

Block-stacking domain and problem

```
(define (domain blocksworld)
(:requirements :strips :equality)
(:predicates (clear ?x)
             (on-table ?x)
             (arm-empty)
             (holding ?x)
             (on ?x ?y))

(:action pickup
  :parameters (?ob)
  :precondition (and (clear ?ob) (on-table ?ob) (arm-empty))
  :effect (and (holding ?ob) (not (clear ?ob)) (not (on-table ?ob))
               (not (arm-empty))))

(:action putdown
  :parameters  (?ob)
  :precondition (and (holding ?ob))
  :effect (and (clear ?ob) (arm-empty) (on-table ?ob)
               (not (holding ?ob))))

(:action stack
  :parameters  (?ob ?underob)
  //:precondition (and  (clear ?underob) (holding ?ob) (not (= ?ob ?underob)) )
  :precondition (and  (clear ?underob) (holding ?ob))
  :effect (and (arm-empty) (clear ?ob) (on ?ob ?underob)
               (not (clear ?underob)) (not (holding ?ob))))

(:action unstack
  :parameters  (?ob ?underob)
  :precondition (and (on ?ob ?underob) (clear ?ob) (arm-empty))
  :effect (and (holding ?ob) (clear ?underob)
(not (on ?ob ?underob)) (not (clear ?ob)) (not (arm-empty)))))
```

```
(define (problem pb3)
   (:domain blocksworld)
   (:objects a b c)
   (:init (on-table a) (on-table b)   (on c a)
          (clear b) (clear c) (arm-empty))
(:goal (and (on a b) (on b c))))
```



How many possible states are there?

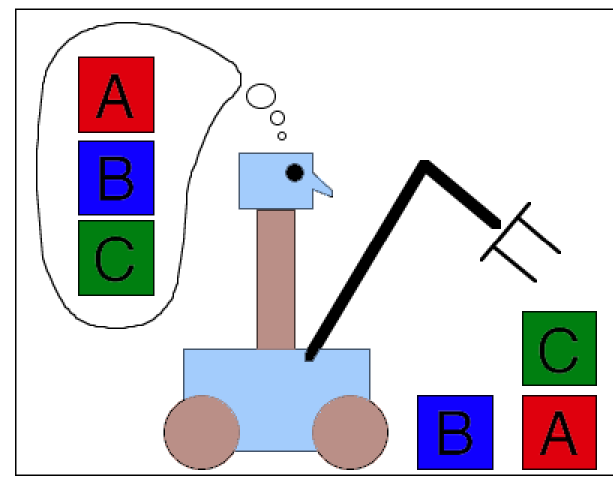
Can we simplify the domain a little?

Block-stacking domain and problem

```
(define (domain blocksworld)
(:requirements :strips :equality)
(:predicates (clear ?x)
             (on-table ?x)
             (arm-empty)
             (holding ?x)
             (on ?x ?y))

(:action pickup
  :parameters (?ob)
  :precondition (and (clear ?ob) (on-table ?ob) (arm-empty))
  :effect (and (holding ?ob) (not (clear ?ob)) (not (on-table ?ob))
               (not (arm-empty))))

(:action putdown
  :parameters  (?ob)
  :precondition (and (holding ?ob))
  :effect (and (clear ?ob) (arm-empty) (on-table ?ob)
               (not (holding ?ob))))

(:action stack
  :parameters  (?ob ?underob)
  //:precondition (and  (clear ?underob) (holding ?ob) (not (= ?ob ?underob)) )
  :precondition (and  (clear ?underob) (holding ?ob))
  :effect (and (arm


(:action unstac
  :parameters
  :precondition
  :effect (and
(not (on ?ob ?undero
```

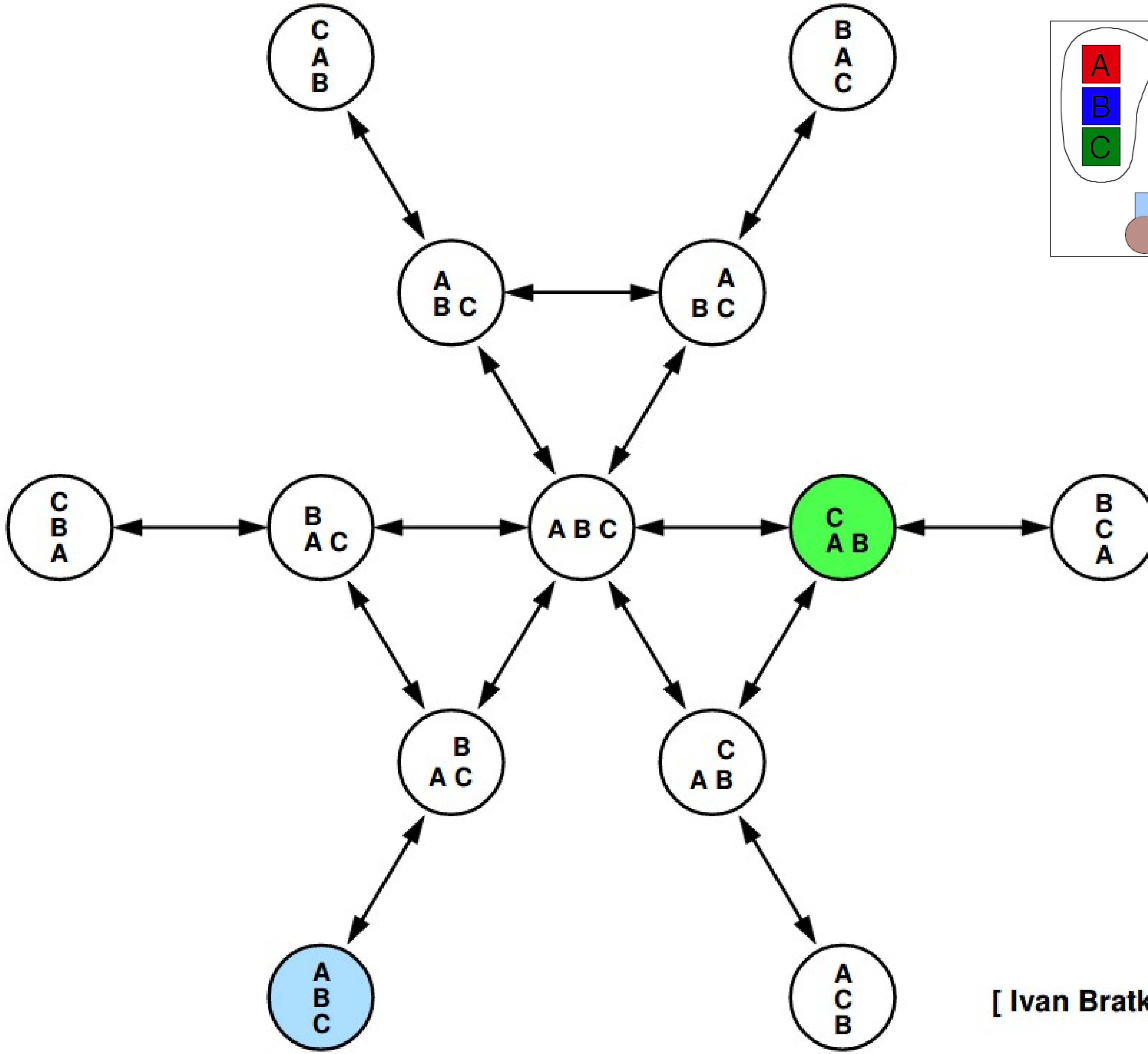
```
(define (problem pb3)
   (:domain blocksworld)
   (:objects a b c)
   (:init (on-table a) (on-table b)   (on c a)
          (clear b) (clear c) (arm-empty))
(:goal (and (on a b) (on b c))))
```

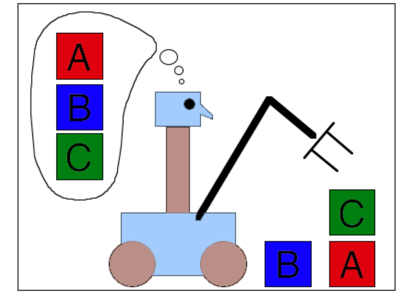
Send me the simpler PDDL domain and problem for extra credit!



How many possible states are there?

Can we simplify the domain a little?





[Ivan Bratko]

Classical Planning Model

Planning with **deterministic** actions under **complete knowledge**

Characterized by:

- a finite **state space** S
- a finite set of **actions** A; $A(s)$ are actions **executable** at s
- **deterministic** transition function $f : S \times A \rightarrow S$ such that $f(s, a)$ is state after applying action $a \in A(s)$ in state s
- **known** initial state s_{init}
- subset $G \subseteq S$ of **goal states**
- **positive costs** $c(s, a)$ of applying action a in state s

(often, $c(s, a)$ only depends on a)

Classical Planning Model

Since the initial state is **known** and the effects of the actions can be **predicted**, a controller is a **fixed** action sequence $\pi = \langle a_0, a_1, \ldots, a_n \rangle$

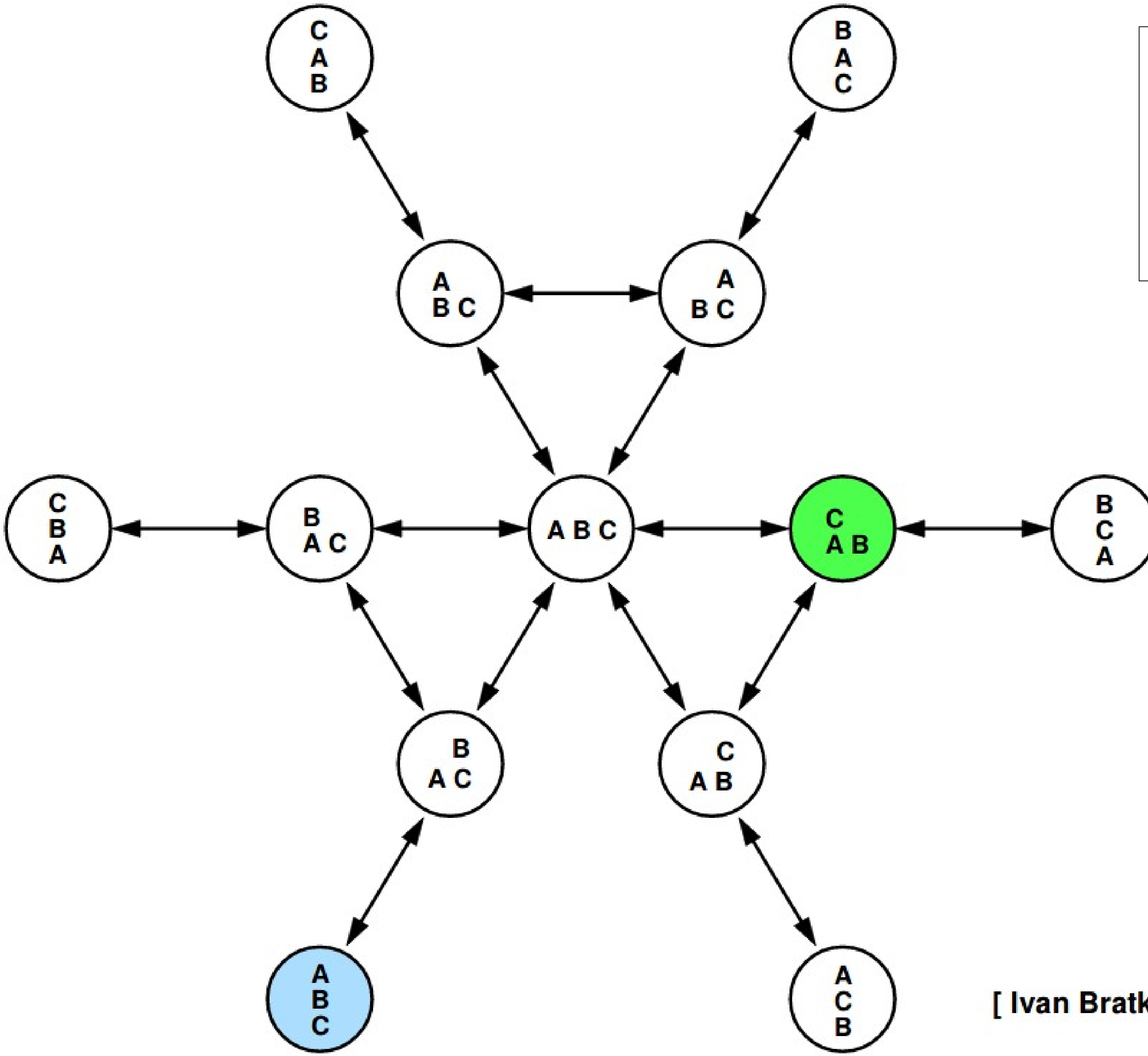
The sequence defines a **state trajectory** $\langle s_0, s_1, \ldots, s_{n+1} \rangle$ where:

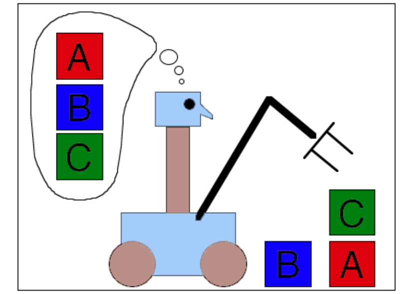
- $s_0 = s_{init}$ is the initial state
- $a_i \in A(s_i)$ is an applicable action at state s_i, $i = 0, \ldots, n$
- $s_{i+1} = f(s_i, a_i)$ is the result of applying action a_i at state s_i

The controller is **valid** (i.e., solution) iff s_{n+1} is a goal state

Its **cost** is $c(\pi) = c(s_0, a_0) + c(s_1, a_1) + \cdots + c(s_n, a_n)$

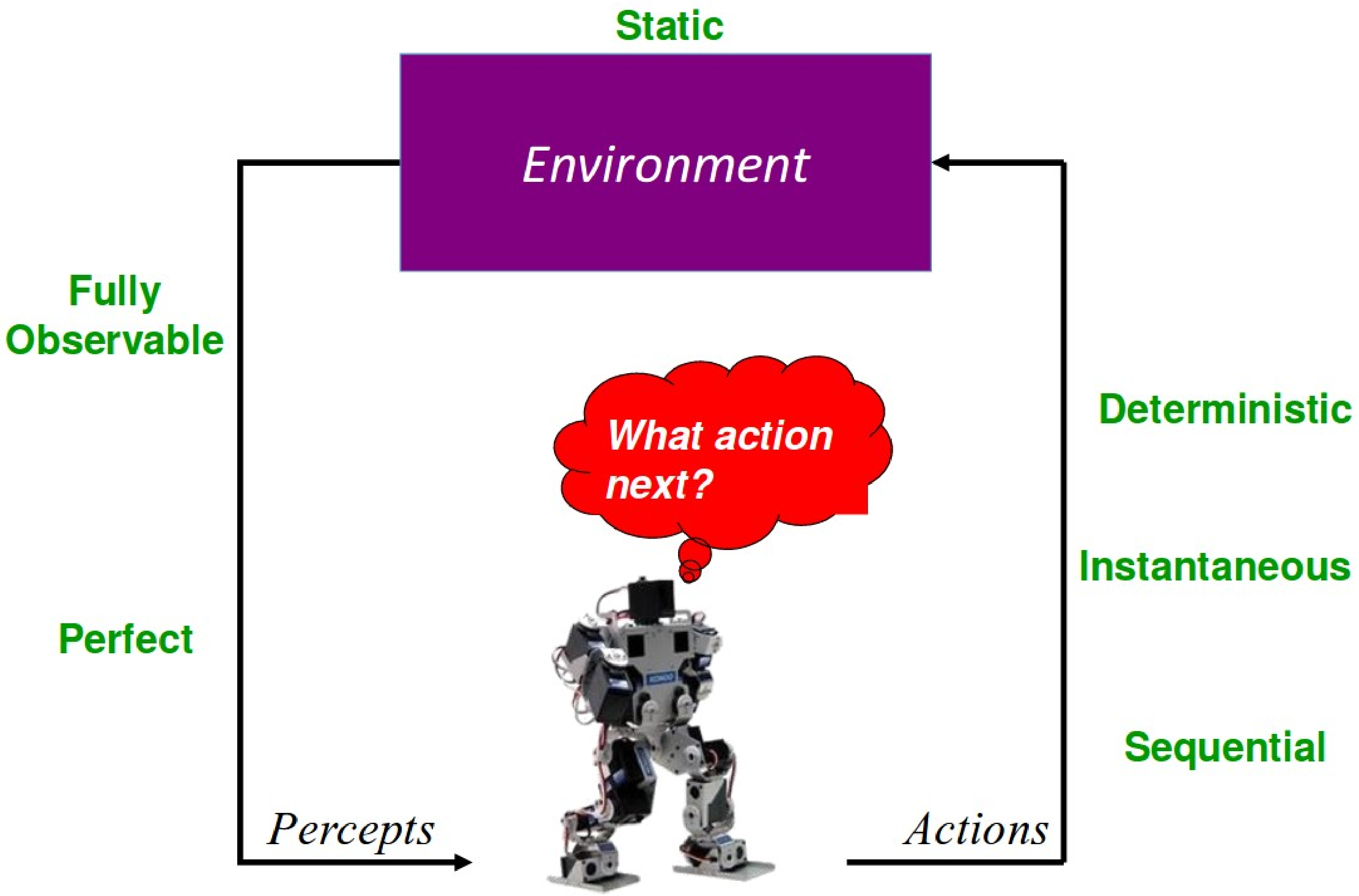
It is **optimal** if its cost is minimum among all solutions



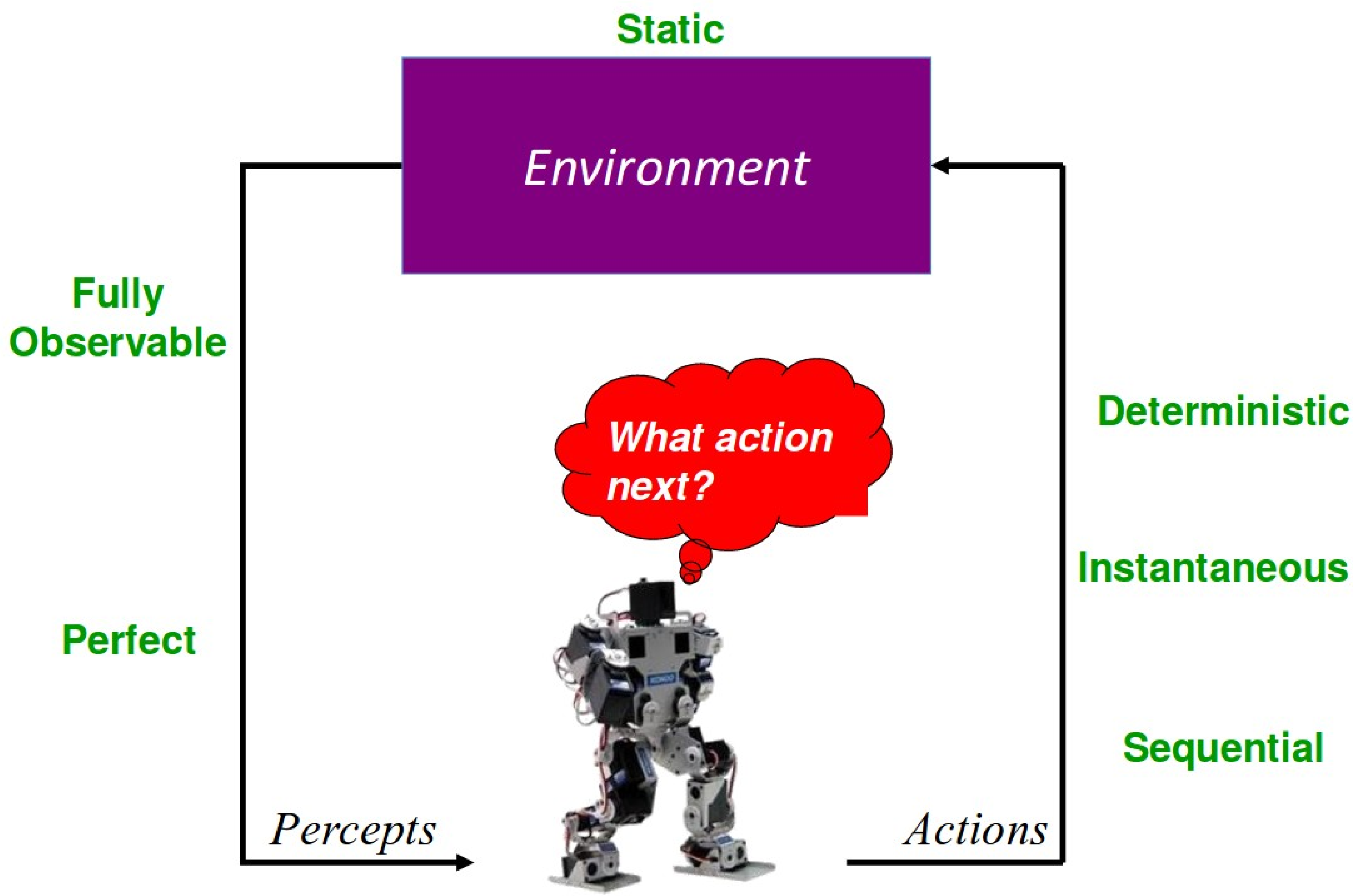


[Ivan Bratko]

Classical Planning

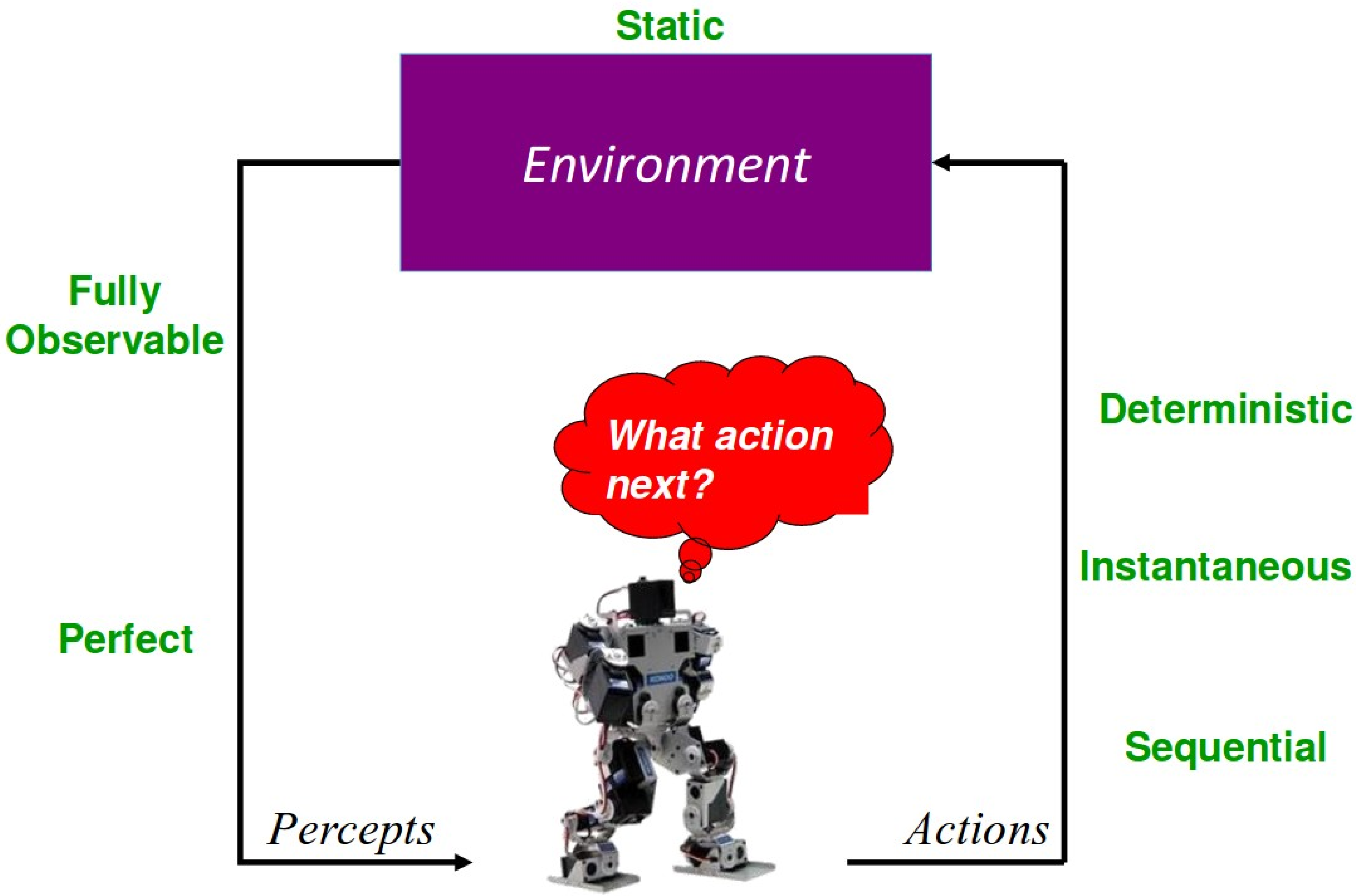


Classical Planning



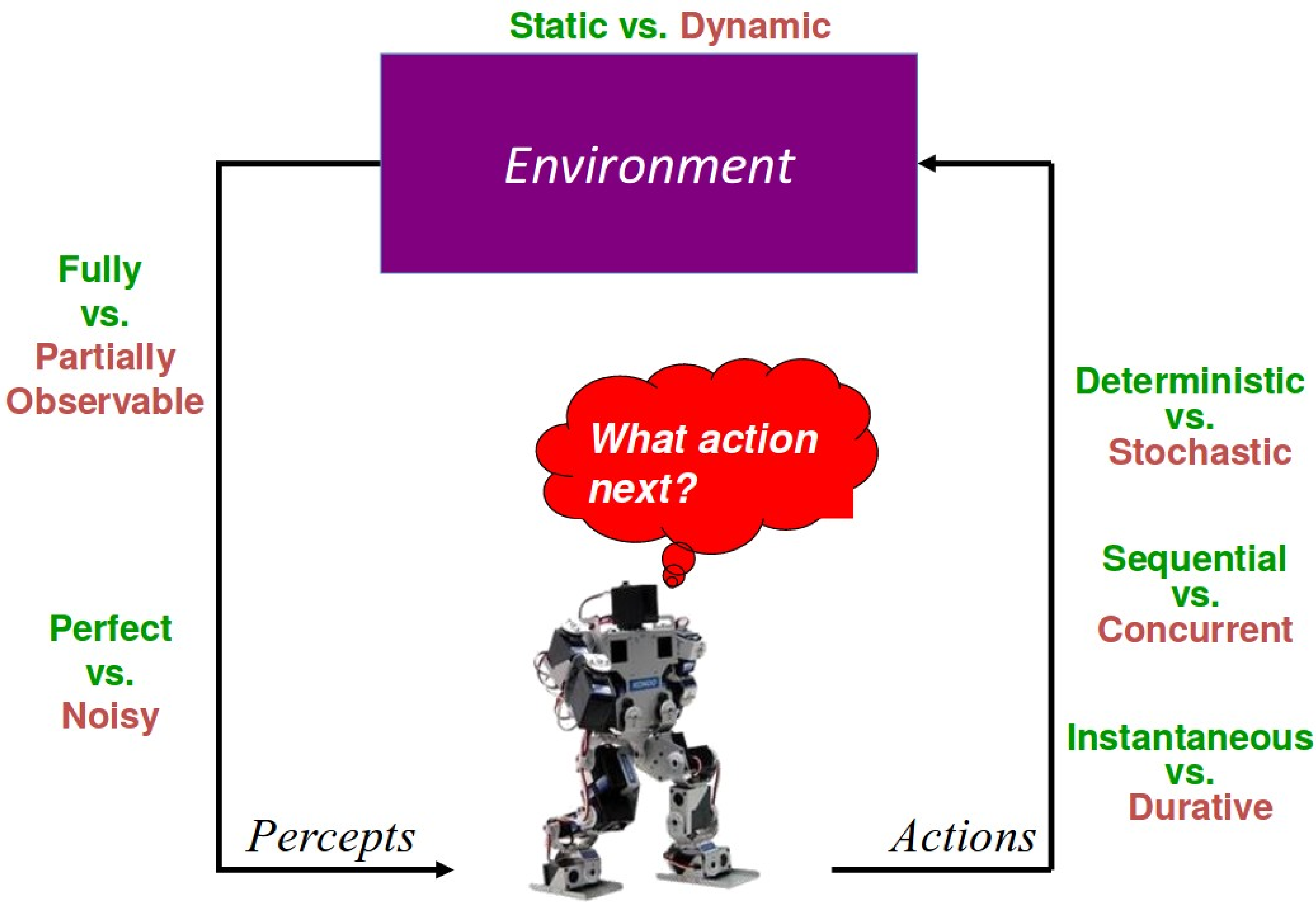
What do we mean by each of these words?

Classical Planning

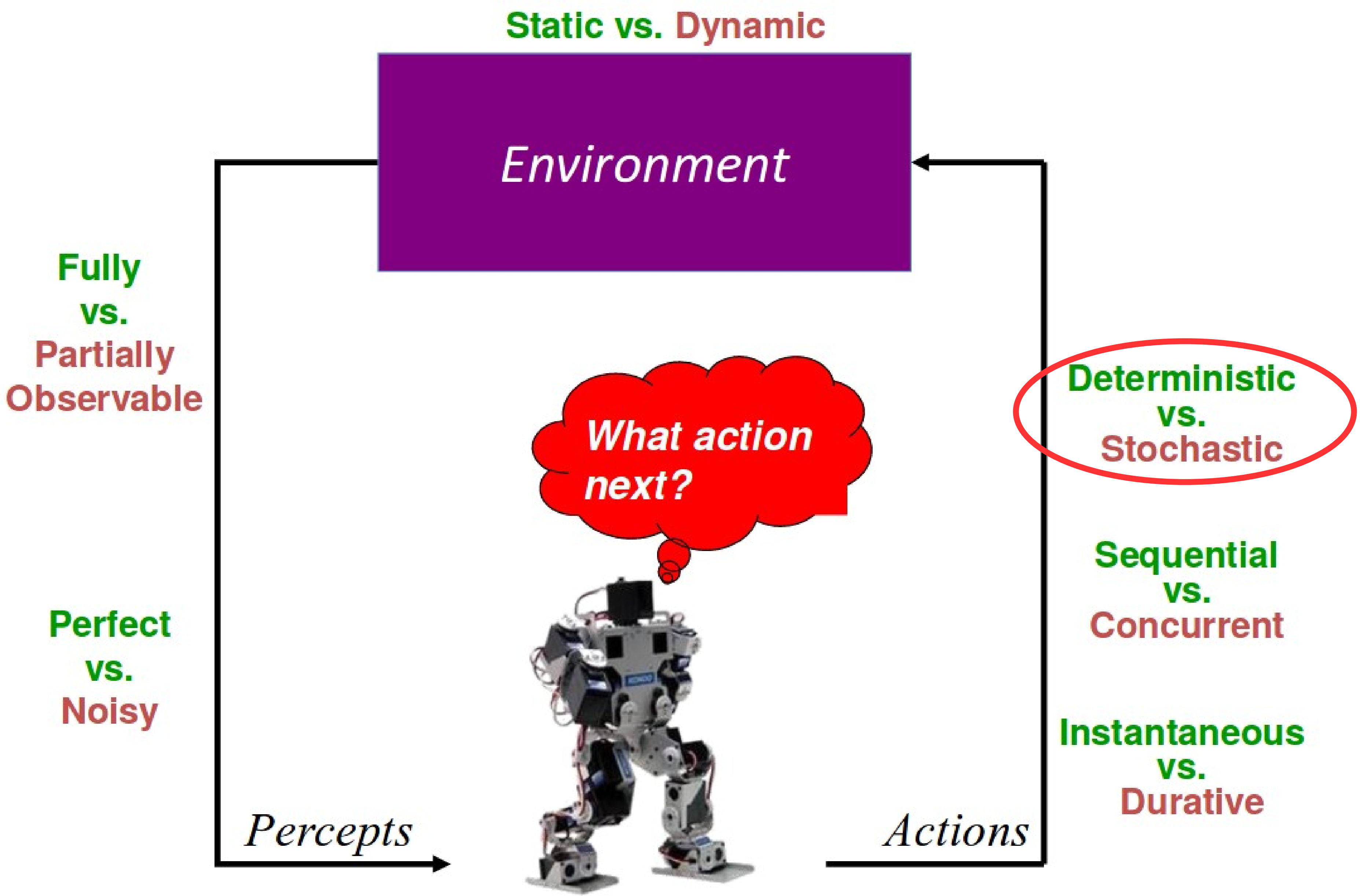


What are some alternatives to these assumptions?

Planning


Static vs. Dynamic
Environment
Fully vs. Partially Observable
Perfect vs. Noisy
What action next?
Deterministic vs. Stochastic
Sequential vs. Concurrent
Instantaneous vs. Durative
Percepts
Actions

Planning


Static vs. Dynamic
Environment
Fully vs. Partially Observable
Perfect vs. Noisy
What action next?
Deterministic vs. Stochastic
Sequential vs. Concurrent
Instantaneous vs. Durative
Percepts
Actions

Actions with Uncertain Effects

- Certain problems have actions whose behaviour is **non-deterministic**

  E.g., tossing a coin or rolling a dice are actions whose outcomes cannot be predicted with certainty

- In other cases, uncertainty is the result of a **coarse model** that doesn't include all the information required to predict the outcomes of actions

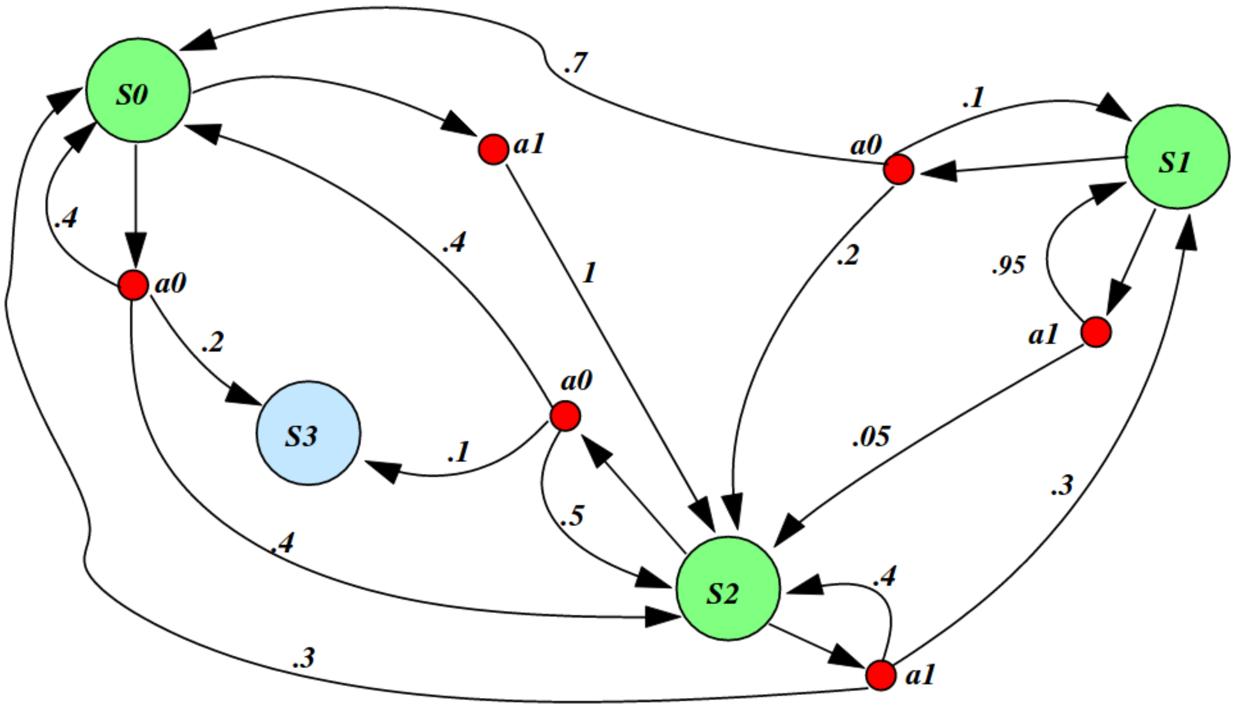
In both cases, it is necessary to consider problems with non-deterministic actions

Mathematical Models of Probabilistic Planning

- A finite state space S
- a finite set of actions A; $A(s)$ are actions executable at sS
- **stochastic** transitions given by **distributions** $p(\cdot|s,a)$ where $p(s'|s,a)$ is the probability of reaching s' when a is executed at s
- initial state s_{init}
- subset $G \subseteq S$ of goal states
- positive costs $c(s,a)$ of applying action a in state s

States are assumed to be fully observable

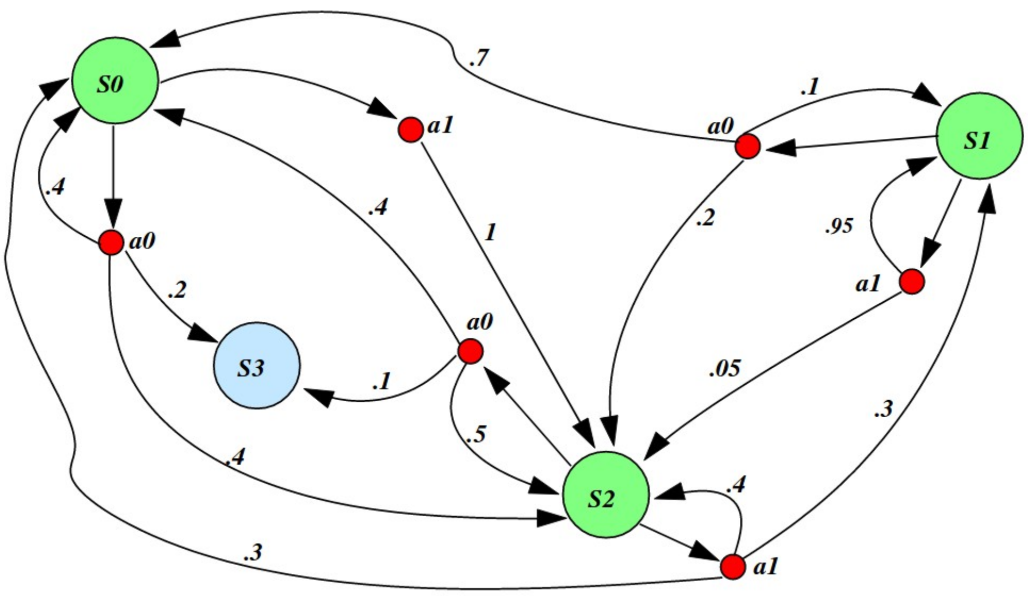
A simple problem

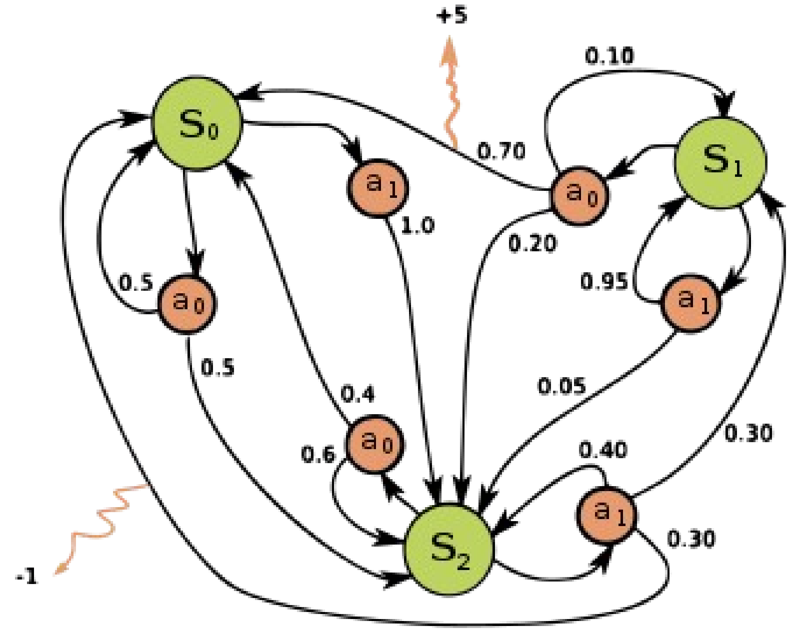


- 4 states; $S = \{s_0, \ldots, s_3\}$
- 2 actions; $A = \{a_0, a_1\}$
- 1 goal; $G = \{s_3\}$

- $p(s_2|s_0, a_1) = 1.0$
- $p(s_0|s_1, a_0) = 0.7$
- $p(s_2|s_2, a_1) = 0.4$

Relation to Markov Decision Processed (MDPs)





Continue on to ICAPS tutorial...

Credits

Andrey Kolobov
Microsoft Research

Alan Fern
Oregon State EECS