Learning for Long-horizon Planning

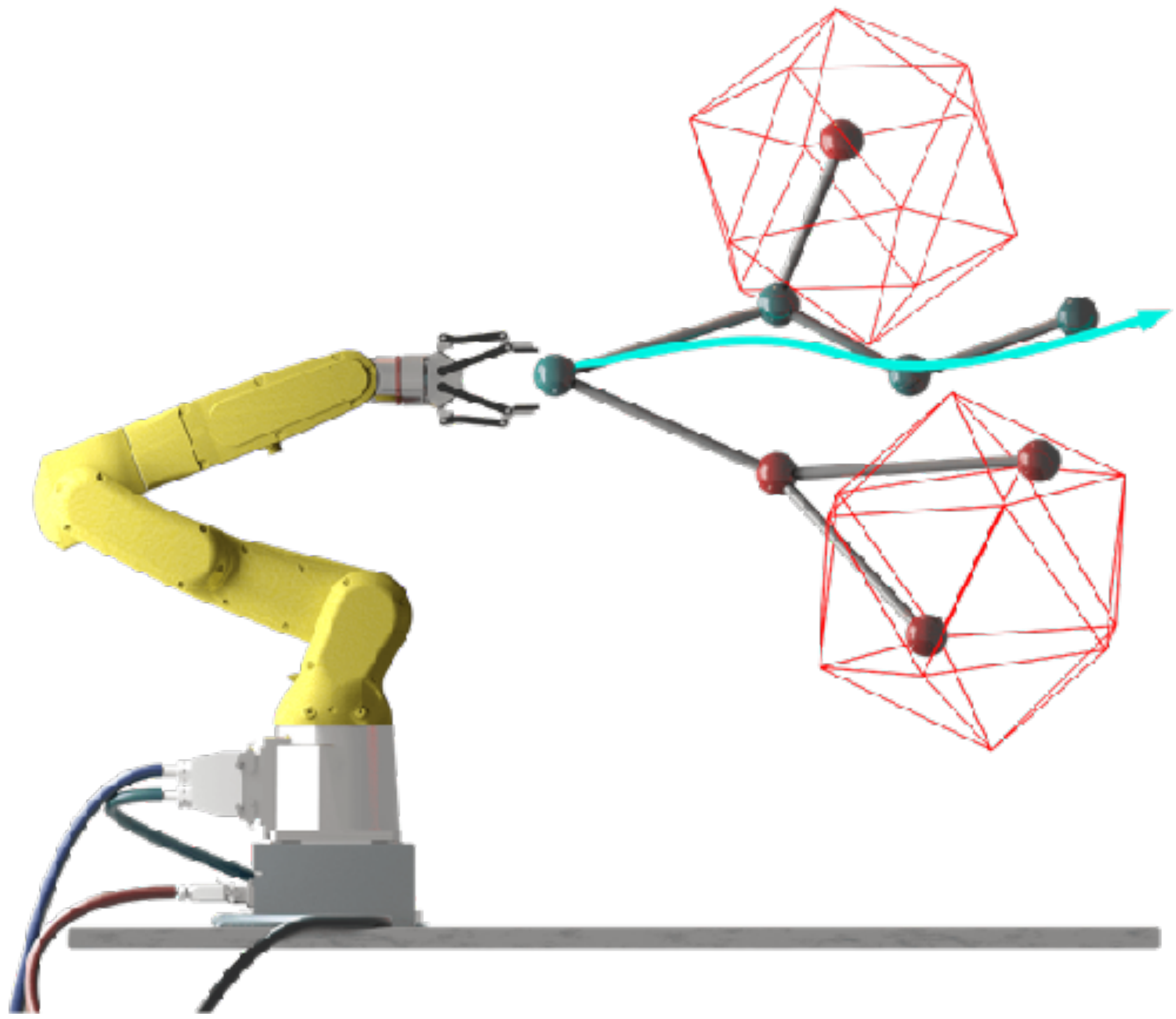
CS 5180 Reinforcement Learning / Guest Lecture

Linfeng Zhao

April 10, 2024

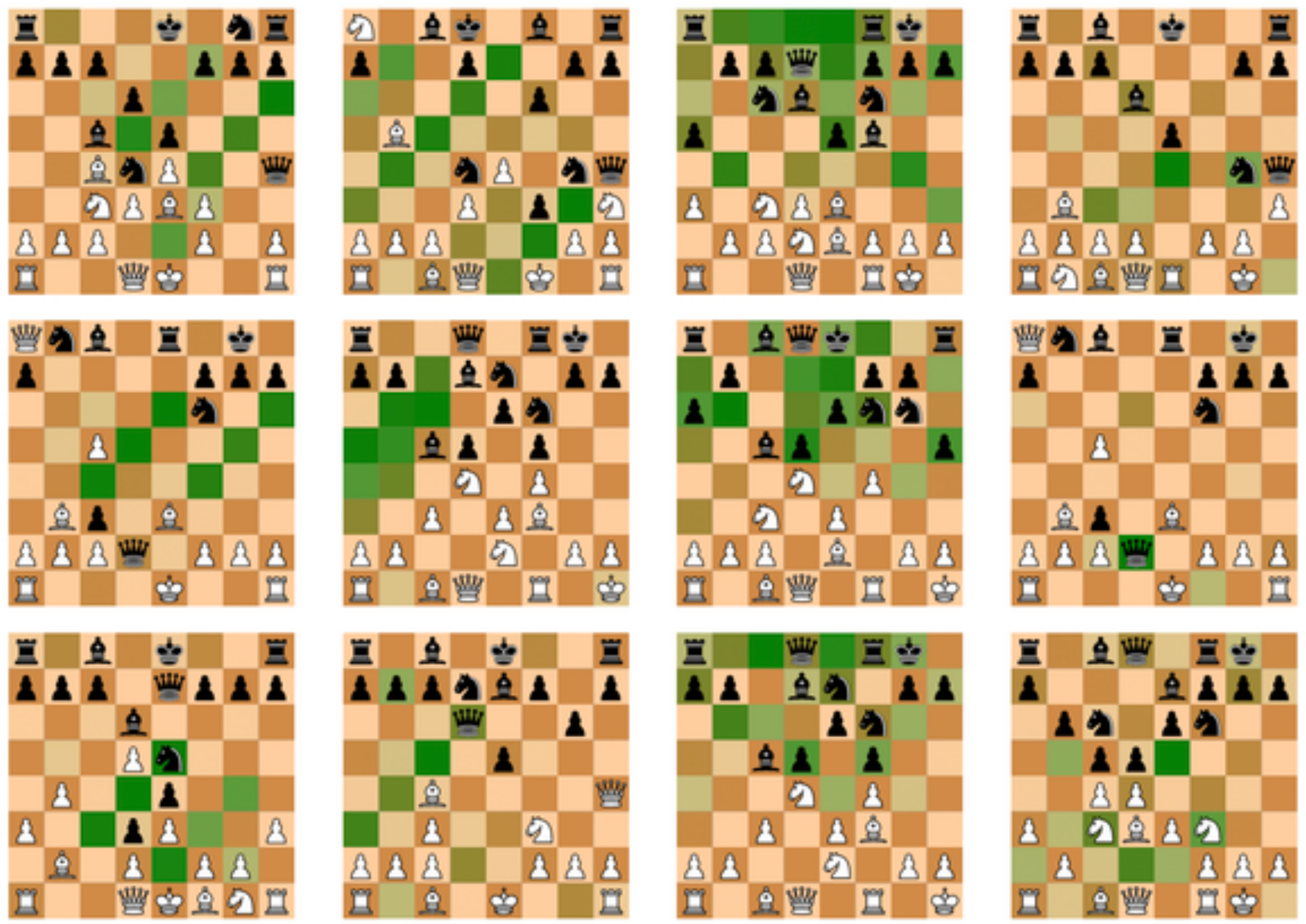
Motivation

You might have used planning in

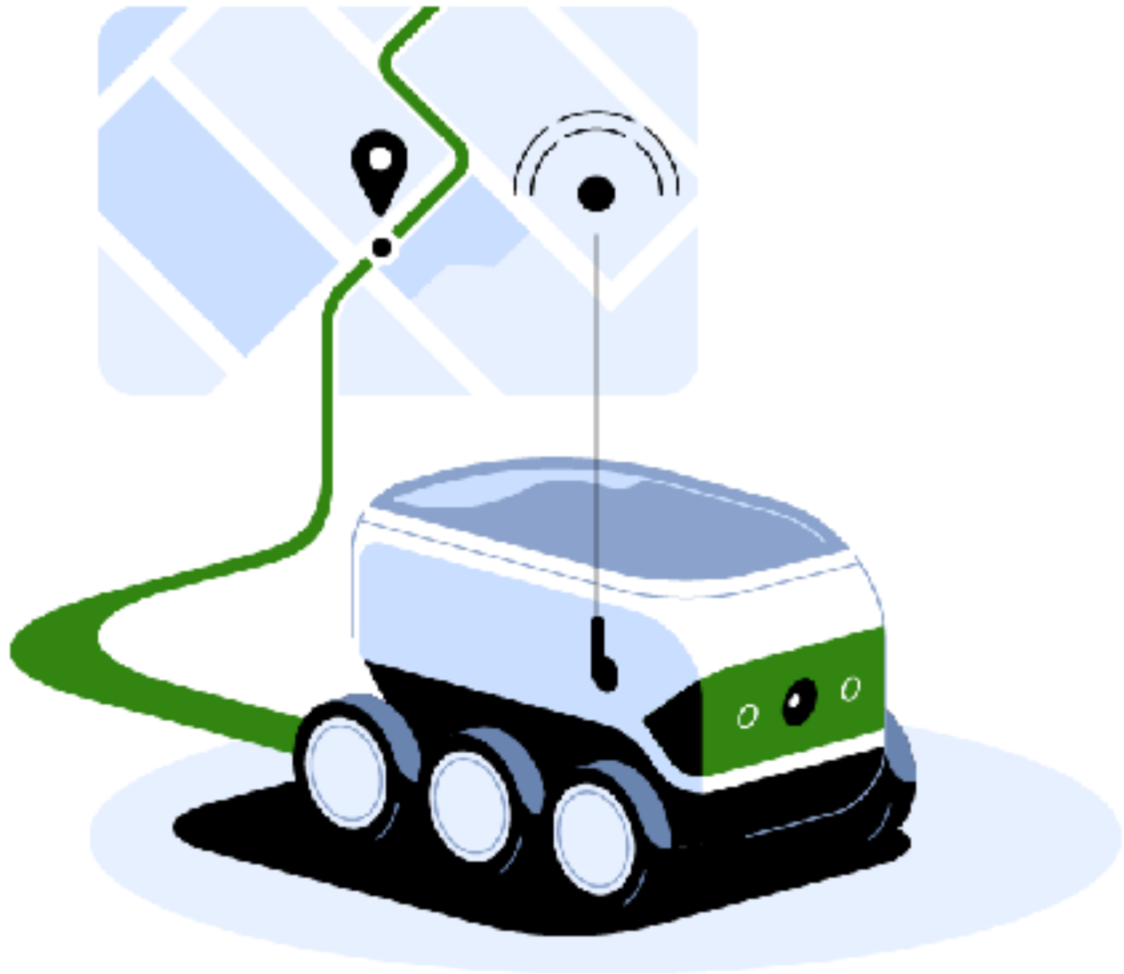


Robot Manipulation / Control Arm

Board games, e.g., Chess



(a) Development of diagonal moves for player (block 1, factor 26 of 36).
(b) Fully developed diagonal moves for opponent (block 3, factor 22 of 36).
(c) Count of opponent's potential piece moves (block 3, factor 11 of 36).
(d) Potential good squares to move to? (block 18, factor 22 of 36).



Robot Navigation

Some names related to planning

- path planning
- motion planning
- task planning
- model predictive control
- model-based RL
- …

Example — Go game

Planning can help us find best sequence of actions through search over the action space

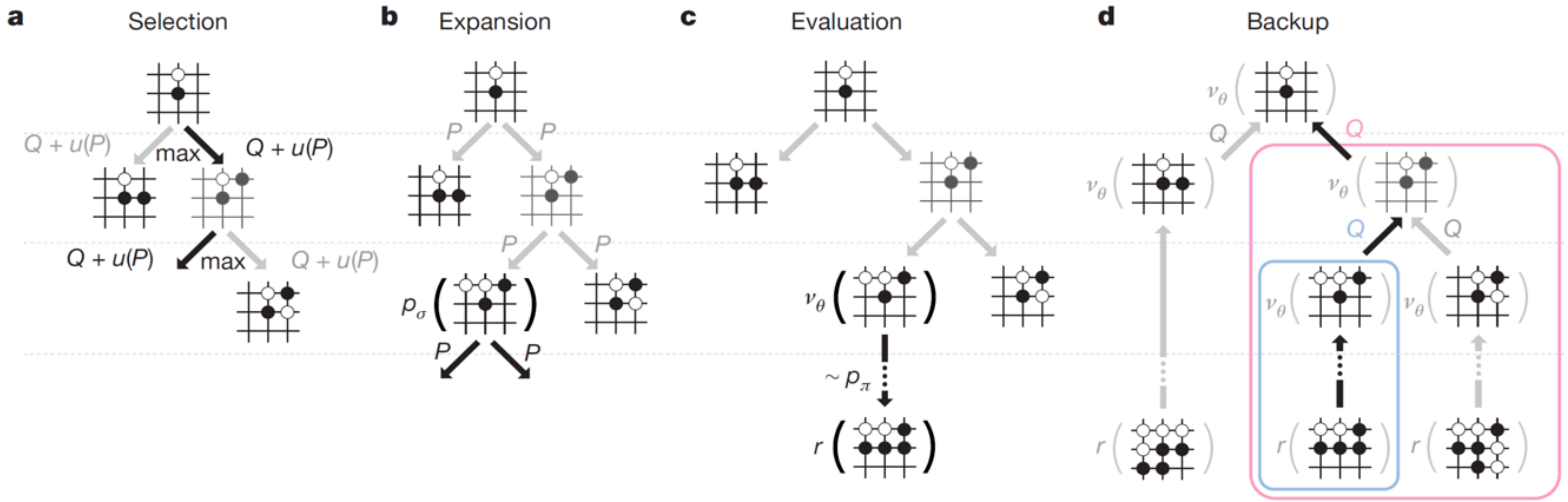
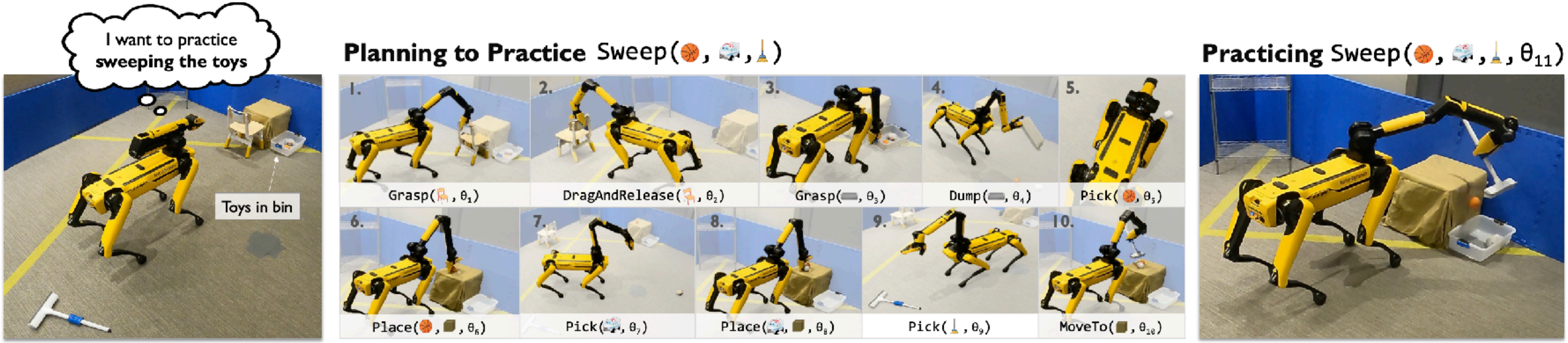


Figure 3 | Monte Carlo tree search in AlphaGo. a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus a bonus $u(P)$ that depends on a stored prior probability P for that edge. **b**, The leaf node may be expanded; the new node is processed once by the policy network p_σ and the output probabilities are stored as prior probabilities P for each action. **c**, At the end of a simulation, the leaf node is evaluated in two ways: using the value network v_θ; and by running a rollout to the end of the game with the fast rollout policy p_π, then computing the winner with function r. **d**, Action values Q are updated to track the mean value of all evaluations $r(\cdot)$ and $v_\theta(\cdot)$ in the subtree below that action.


8X

Example — Mobile Manipulation

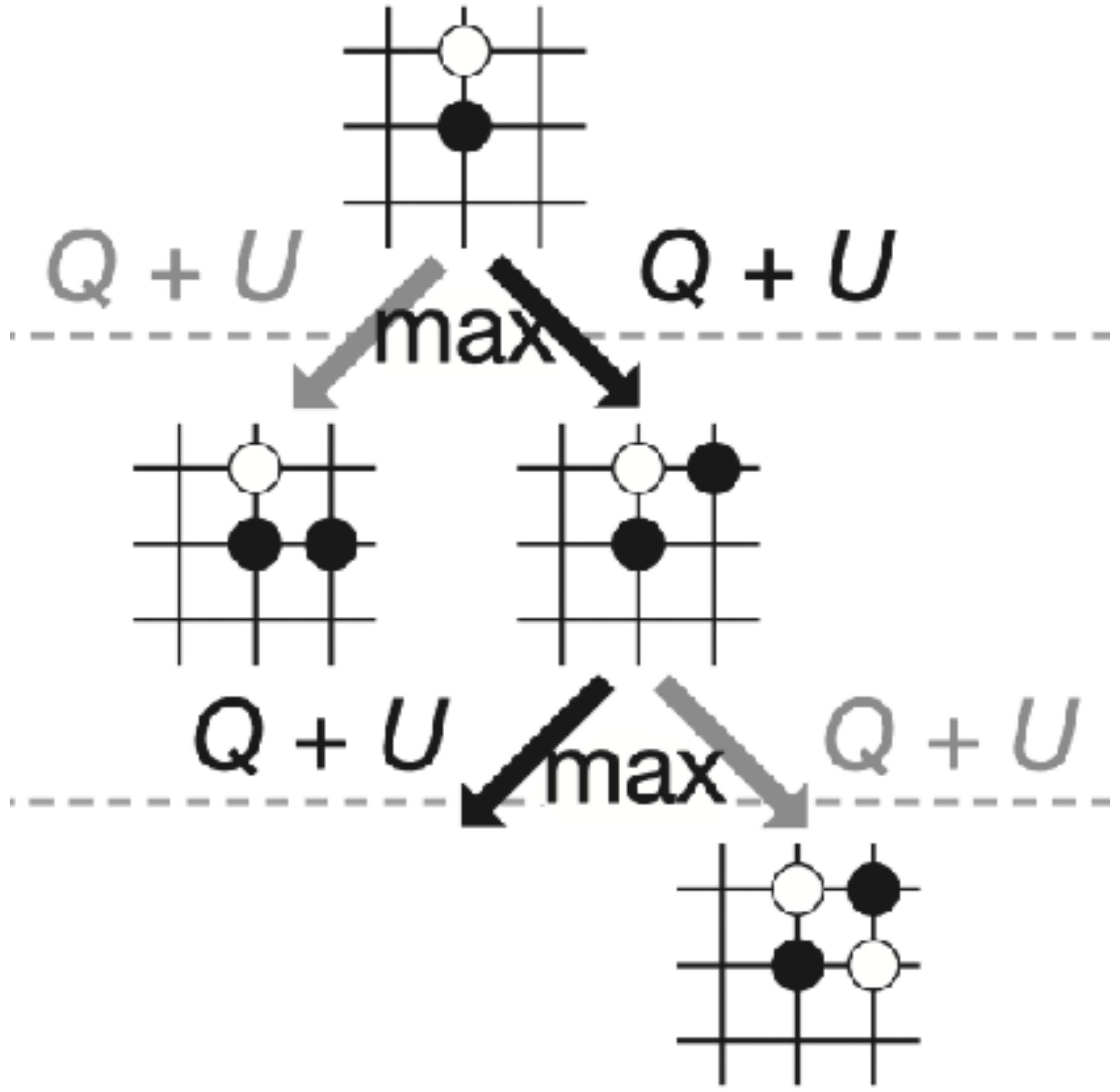


Robot → **Action Sequence**

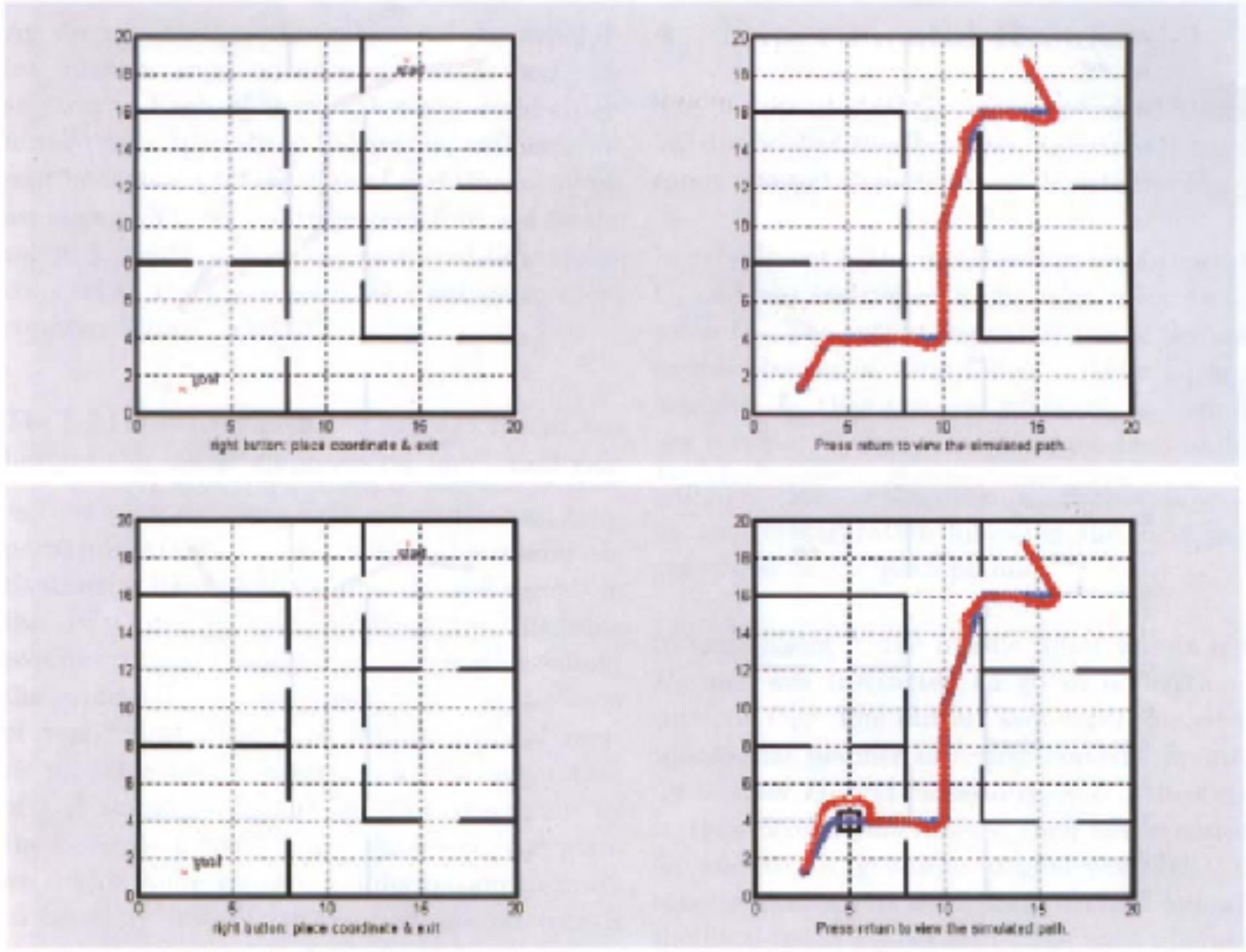
Think over long-horizon

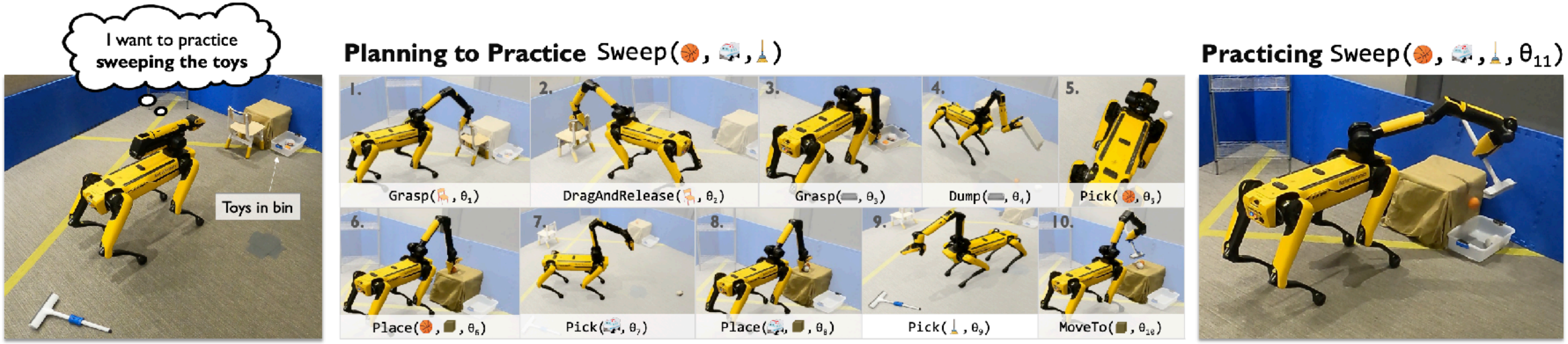
Need planning!

Why do we need planning?



For e.g., solving long-horizon tasks!





Planning is great, but...

Existing planning algorithms normally operate on either **well engineered state features** and **action representations** and are specialized for them

- Discrete: e.g,. graph search, task planner
- Continuous: e.g., model predictive control

However, these features are normally **hand crafted**, which would struggle to scale up

- e.g., discrete graph nodes or continuous vectors

Can planning algorithms handle complex tasks high-dimensional raw input?

Why need learning for planning?

Reason 1 — Approximate complex functions from training data and generalize

- E.g., learning features from raw observations, learning transition functions

Reason 2 — Scaling up with more compute and data → Better planning performance

- ***Scaling laws / The bitter lesson*** — algorithms that can use compute eventually take over
- Learning and Search/Planning are two major types of examples
- Well integration of learning with planning helps planning to **scale up** to more challenging and longer-horizon tasks

(And so on)

Outline

Goals and Motivation

Basics of Planning

The Role of Learning in Planning

Planning Algorithms & Integration with Learning

Case Study: Mobile Manipulation

Takeaways

Basics of Planning

What is planning?

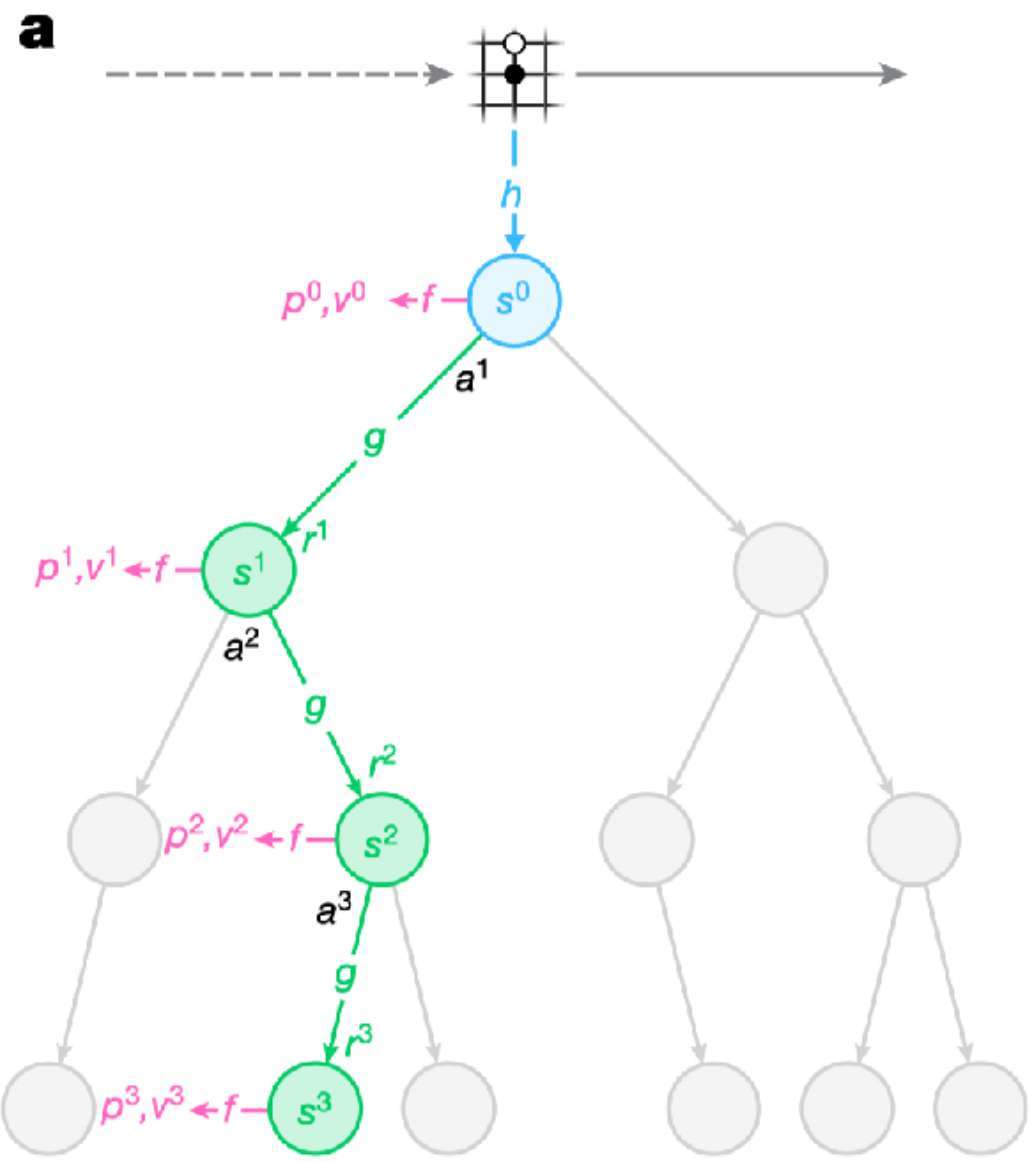
A (learned) state space

A (learned) transition model (world model)

- $M = \langle P, R \rangle$ – $P(s' \mid s, a)$ is transition dynamics, $R(s, a)$ is reward function
- Or deterministic case: $s', r = F(s, a)$

Objective of planning

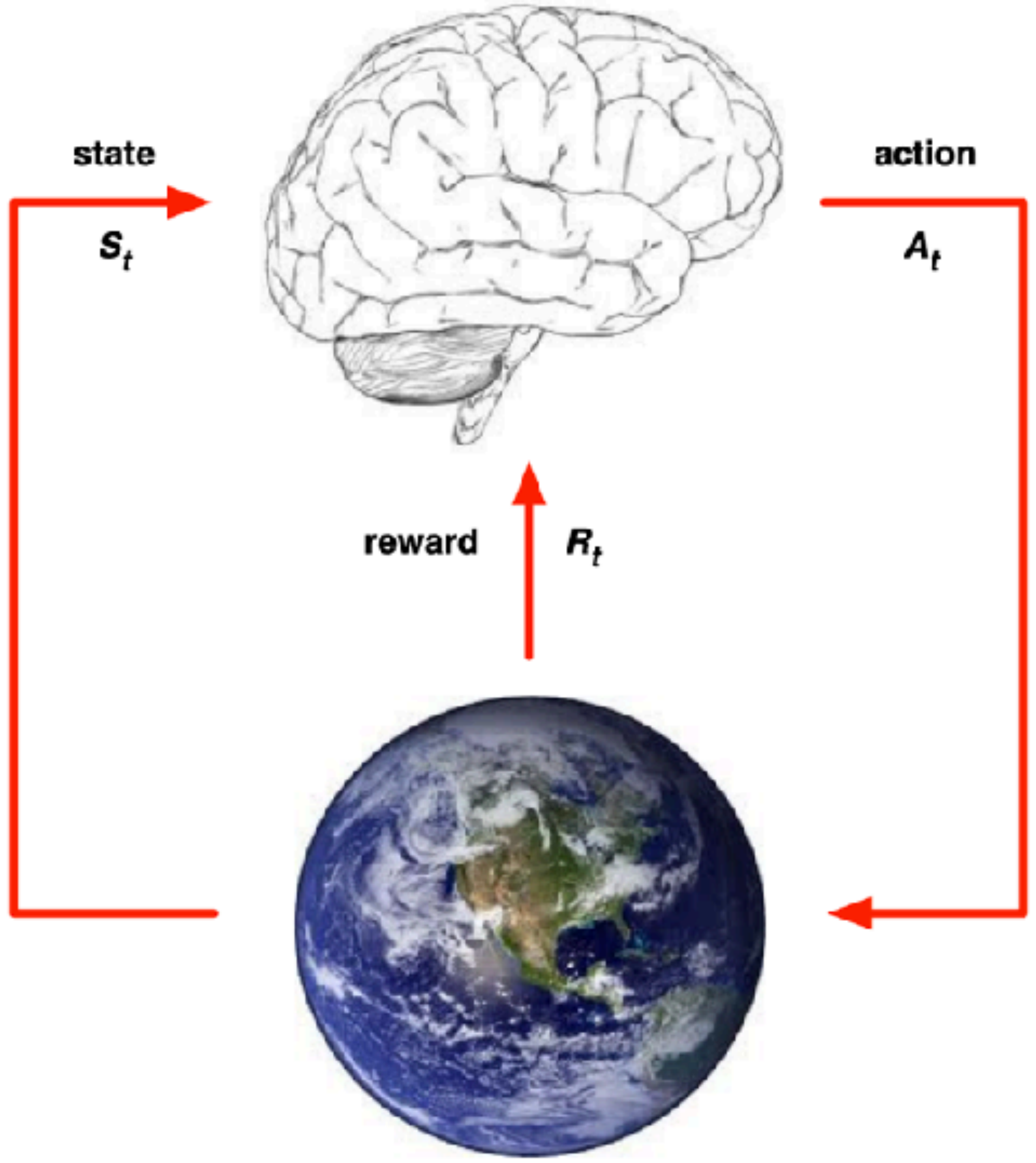
- 1, Maximize reward/utility function, or minimize cost
- 2, Reach goal region



[Credit: MuZero, DeepMind]

1. Planning in RL / Optimal Control

- Model-Free RL
  - No model
  - Learn value function (and/or policy) from experience
- Model-Based RL
  - Learn a model from experience
  - Plan value function (and/or policy) from model



Lecture: Integrated learning and planning. David Silver, 2015.

What is a model here

- A *model* $\mathcal{M}$ is a representation of an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, parametrized by η
- We will assume state space $\mathcal{S}$ and action space $\mathcal{A}$ are known
- So a model $\mathcal{M} = \langle \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$ represents state transitions $\mathcal{P}_\eta \approx \mathcal{P}$ and rewards $\mathcal{R}_\eta \approx \mathcal{R}$

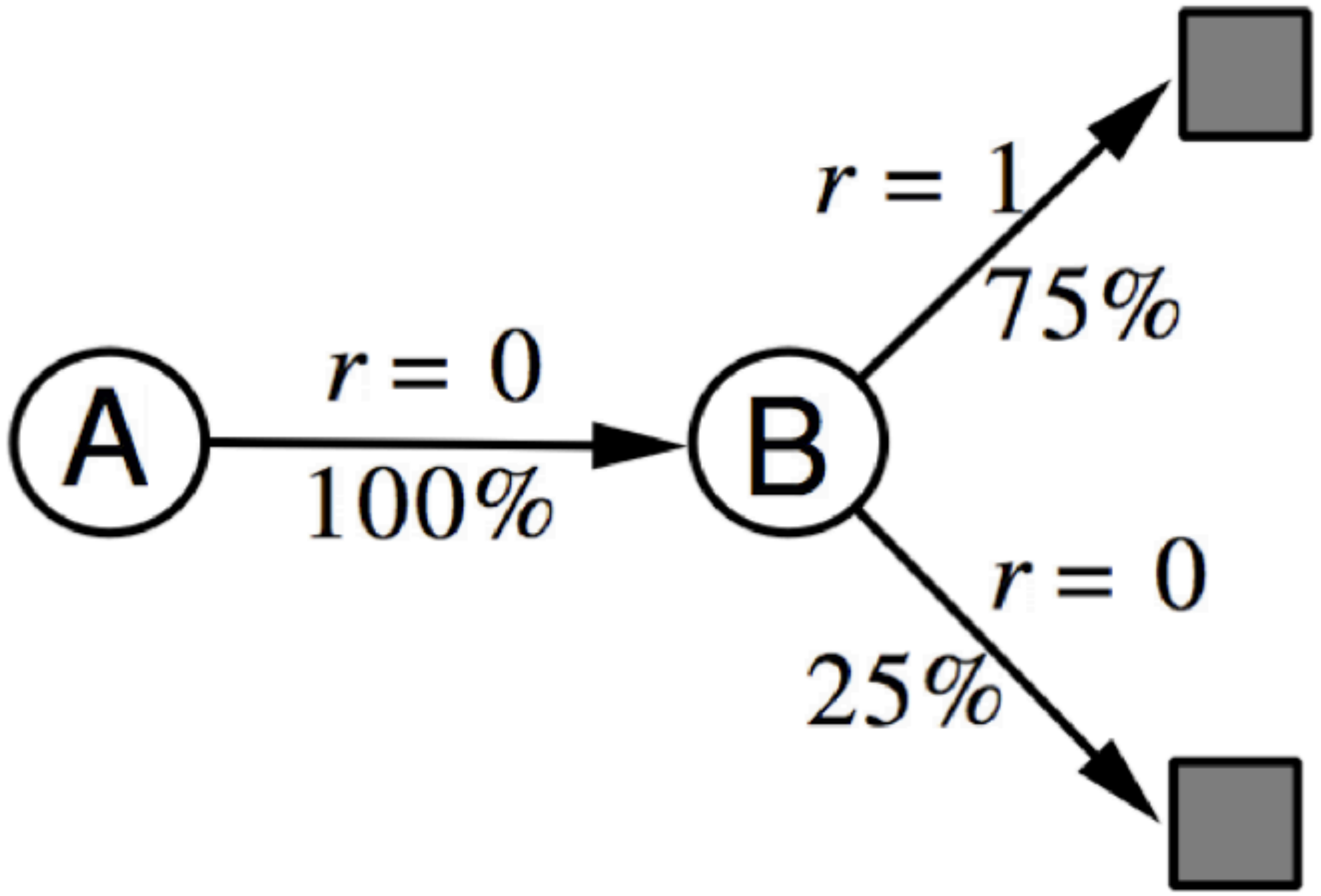
$$S_{t+1} \sim \mathcal{P}_\eta(S_{t+1} \mid S_t, A_t)$$
$$R_{t+1} = \mathcal{R}_\eta(R_{t+1} \mid S_t, A_t)$$

Lecture: Integrated learning and planning. David Silver, 2015.

Example: Learning a table lookup model

Two states A, B; no discounting; 8 episodes of experience

$A, 0, B, 0$
$B, 1$
$B, 1$
$B, 1$
$B, 1$
$B, 1$
$B, 1$
$B, 0$

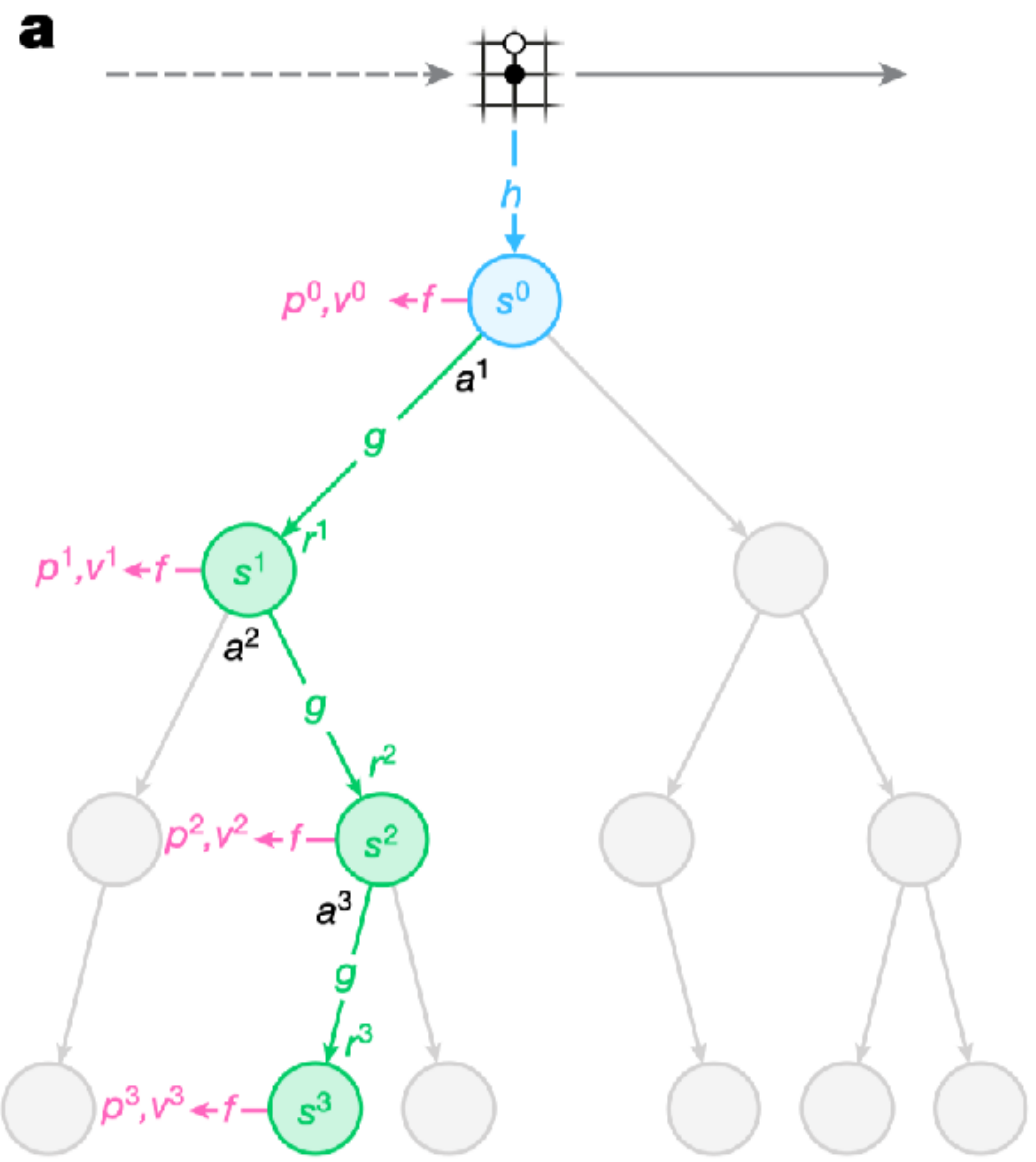


We have constructed a table lookup model from the experience

Lecture: Integrated learning and planning. David Silver, 2015.

Planning with the learned model

- Given a model $\mathcal{M}_\eta = \langle \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$
- Solve the MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$
- Using favourite planning algorithm
  - Value iteration
  - Policy iteration
  - Tree search
  - ...



[Credit: MuZero, DeepMind]

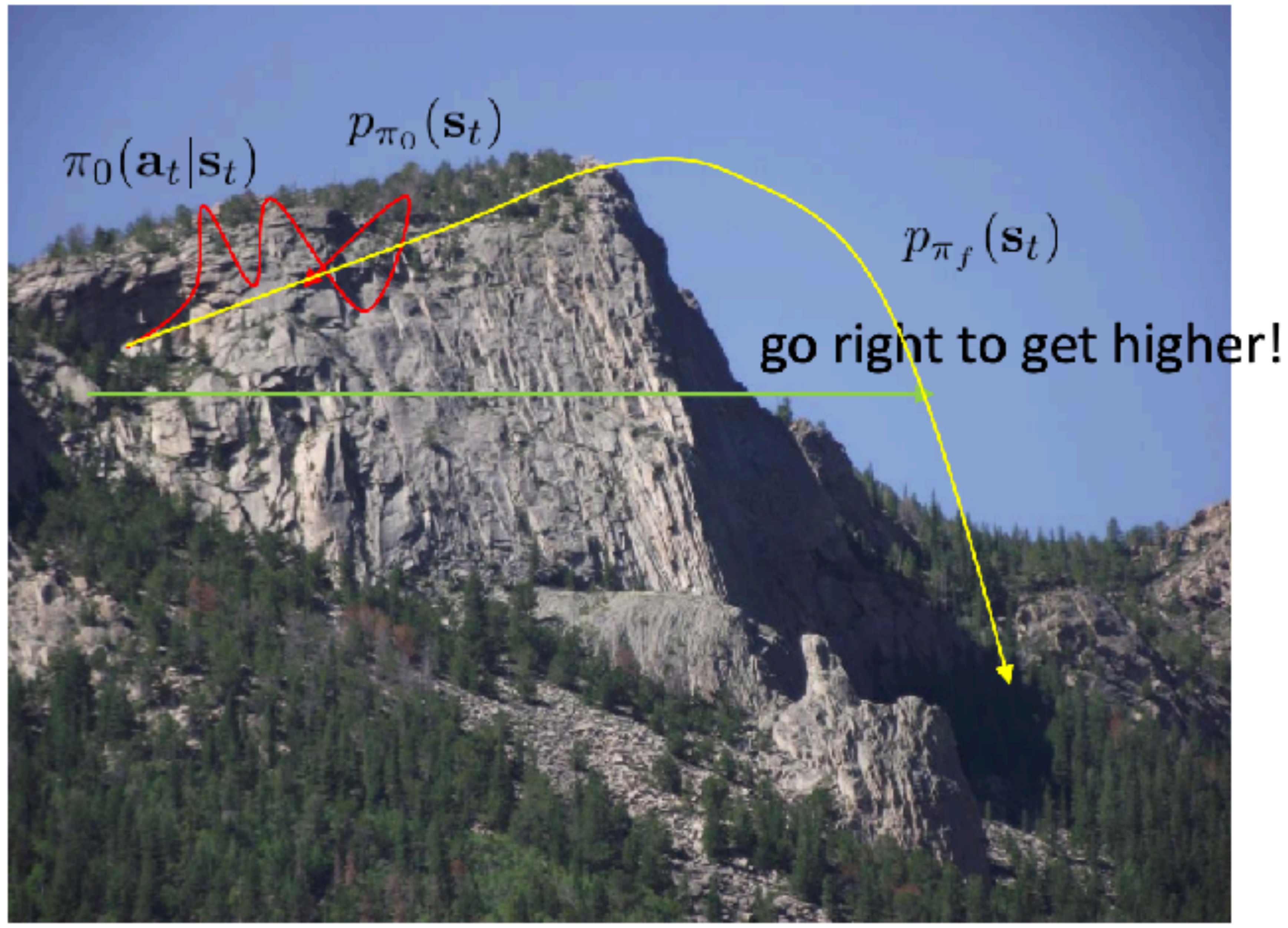
Lecture: Integrated learning and planning. David Silver, 2015.

Does it work? Yes!

- Essentially how system identification works in classical robotics
- Some care should be taken to design a good base policy
- Particularly effective if we can hand-engineer a dynamics representation using our knowledge of physics, and fit just a few parameters

Lecture: Optimal Control and Planning & Model-based RL. Sergey Levine, 2017.

Does it work? No!



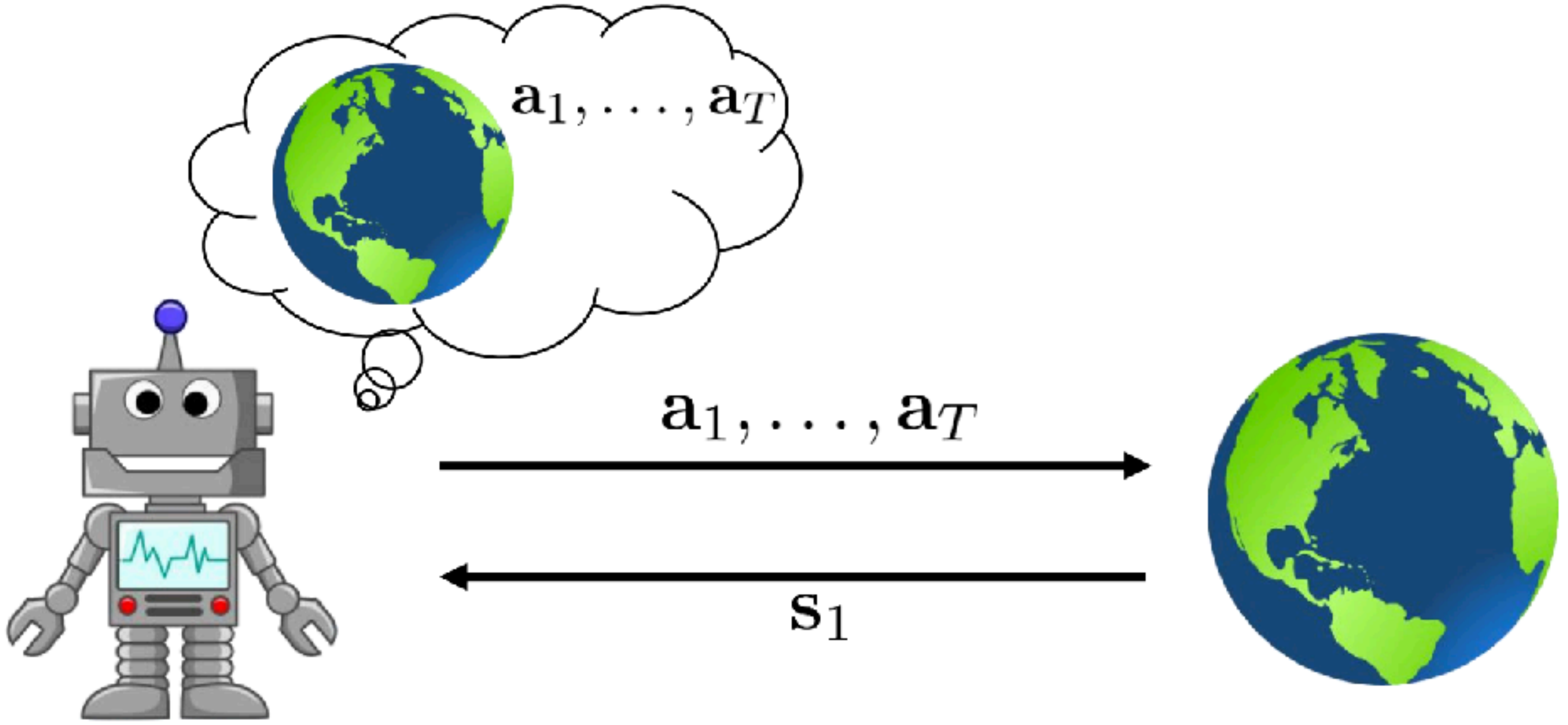
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions

$$p_{\pi_f}(\mathbf{s}_t) \neq p_{\pi_0}(\mathbf{s}_t)$$

- **Distribution mismatch problem becomes exacerbated as we use more expressive model classes**

Lecture: Optimal Control and Planning & Model-based RL. Sergey Levine, 2017.

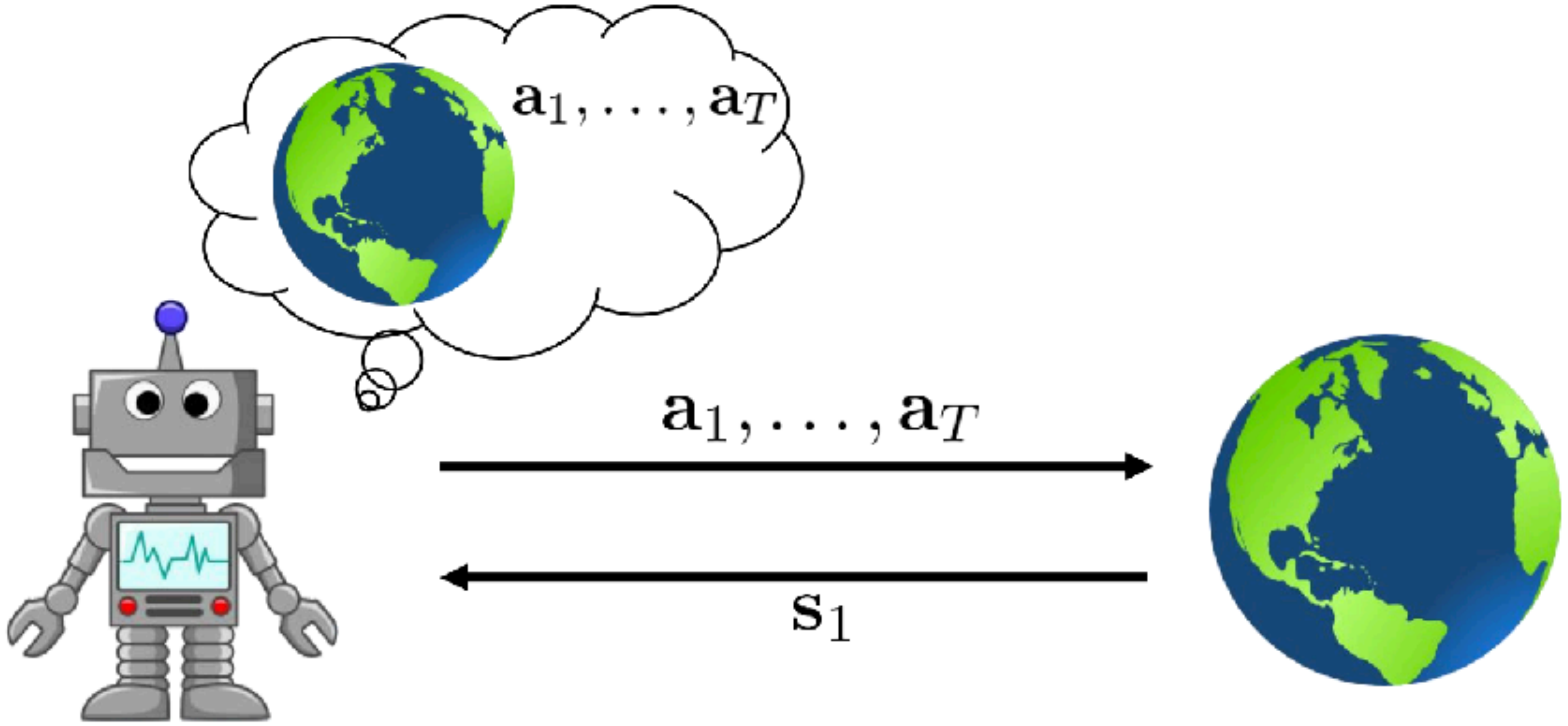
The deterministic case



$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg\max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \text{ s.t. } \mathbf{a}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$$

Lecture: Optimal Control and Planning & Model-based RL. Sergey Levine, 2017.

The stochastic open-loop case



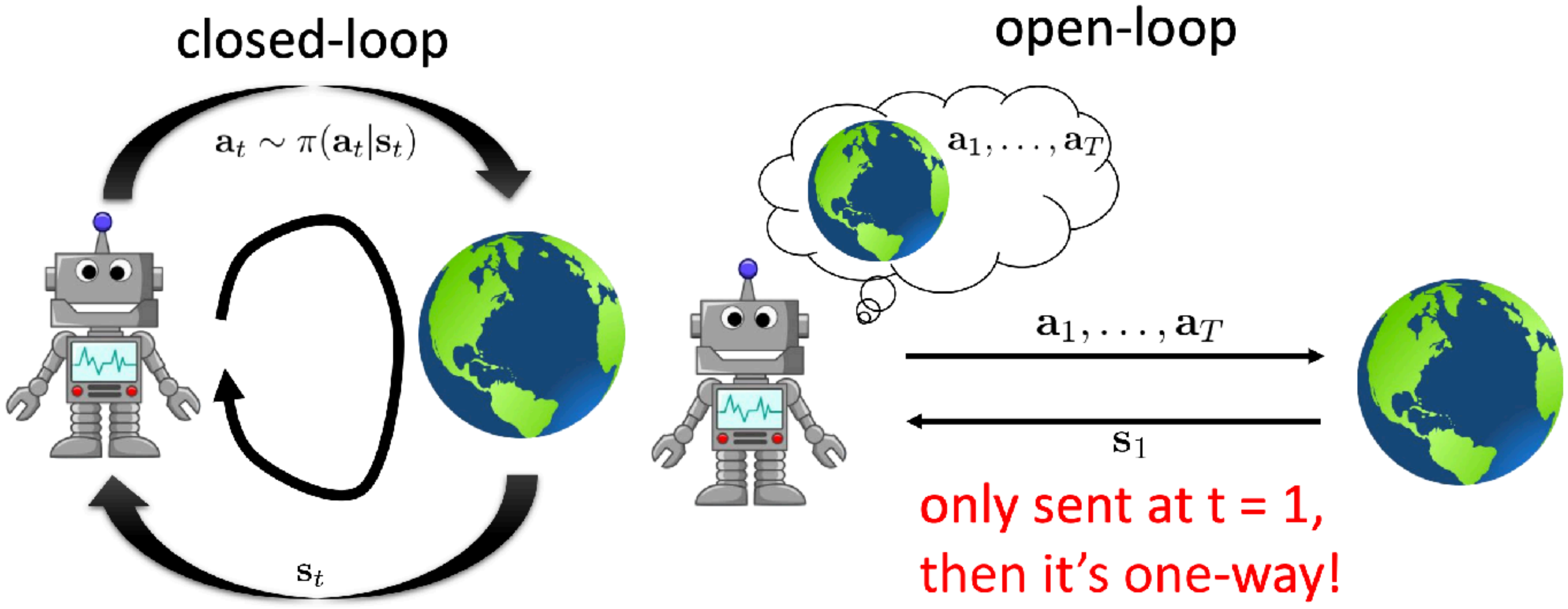
$$p_\theta(\mathbf{s}_1, \ldots, \mathbf{s}_T | \mathbf{a}_1, \ldots, \mathbf{a}_T) = p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg \max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} E \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) | \mathbf{a}_1, \ldots, \mathbf{a}_T \right]$$

Lecture: Optimal Control and Planning & Model-based RL. Sergey Levine, 2017.

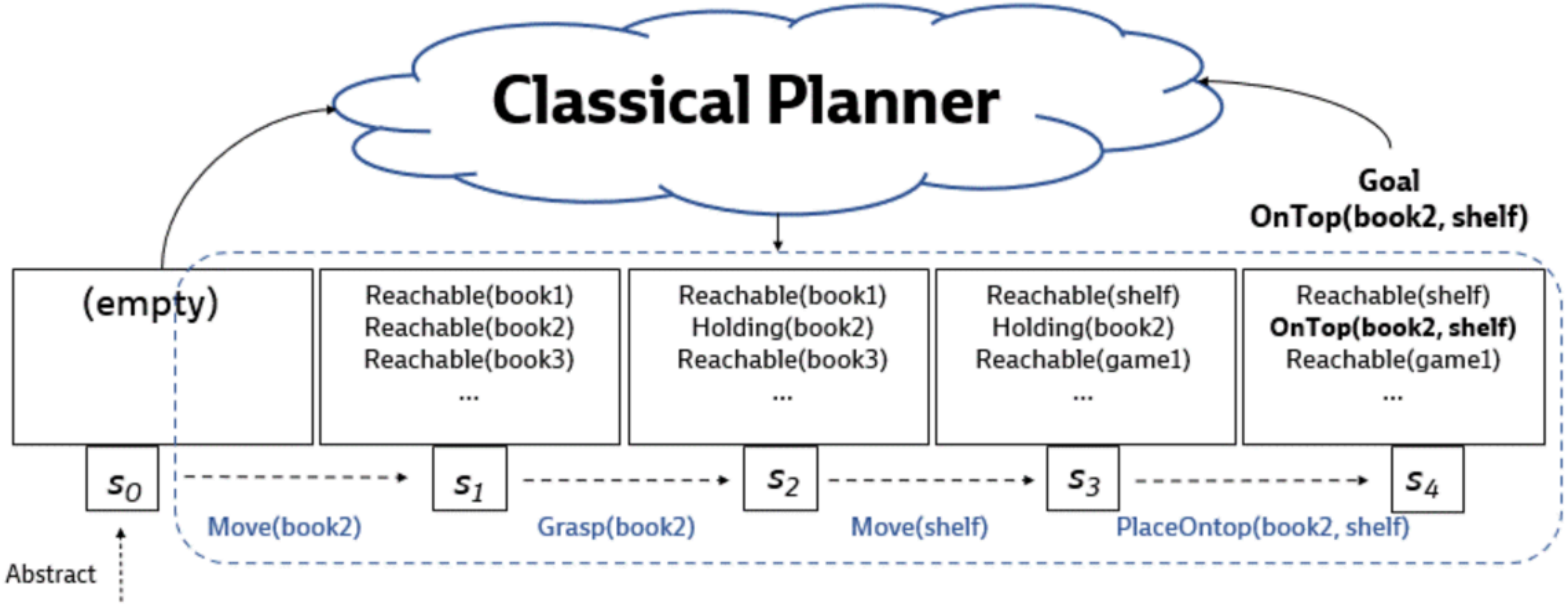
Aside: terminology

what is this "loop"?



Lecture: Optimal Control and Planning & Model-based RL. Sergey Levine, 2017.

2. Planning in classic AI community



AI planners have been developed over decades: STRIPS, PDDL, ...

States are object-centric relational classifiers. Actions are options / operators / skills.

Bilevel Planning for Robots: An Illustrated Introduction. Kumar et al. Blog.

STRIPS / PDDL

Some formal definition.

Connection between options in RL and operators in STRIPS has been studied by George Konidaris:

Konidaris, G., Kaelbling, L. P., & Lozano-Pérez, T. (2018). From skills to symbols: Learning symbolic representations for abstract high-level planning. Journal of Artificial Intelligence Research.

Formulation 2.4 (STRIPS-Like Planning)

1. A finite, nonempty set I of *instances*.
2. A finite, nonempty set P of *predicates*, which are binary-valued (partial) functions of one of more instances. Each application of a predicate to a specific set of instances is called a *positive literal*. A logically negated positive literal is called a *negative literal*.
3. A finite, nonempty set O of *operators*, each of which has: 1) *preconditions*, which are positive or negative literals that must hold for the operator to apply, and 2) *effects*, which are positive or negative literals that are the result of applying the operator.
4. An *initial set* S which is expressed as a set of *positive literals*. Negative literals are implied. For any positive literal that does not appear in S, its corresponding negative literal is assumed to hold initially.
5. A *goal set* G which is expressed as a set of both *positive* and *negative literals*.

Planning Algorithms. Steven M. LaValle, 2006.

Abstraction Actions: Operators

```
Grasp-From-On-Top
    Parameters:      [?target: obj, ?surface: obj]
    Preconditions:   [HandEmpty,
                      Reachable(?target),
                      OnTop(?target, ?surface)]
    Add Effects:     [Holding(?target)]
    Delete Effects:  [HandEmpty,
                      Reachable(?target),
                      OnTop(?target, ?surface)]
    Skill:           Grasp(?target, [Δx, Δy, Δz])
```

Figure 5: Example operator for grasping from atop a surface. The operator has two arguments (both of type 'obj'): a target object to pick up, and a surface from which to pick this object. The operator's associated skill is parameterized by the discrete target object, as well as three continuous parameters that correspond to a cartesian position in the object's coordinate frame at which the robot should attempt to grasp the object..

Bilevel Planning for Robots: An Illustrated Introduction. Kumar et al. Blog.

Connection of 2 formalisms

Connection:

The MDP (state-space representation) in RL and STRIPS/PDDL in classic AI planning are **closely connected**

A STRIPS/PDDL task planning problem can be converted to state-space representation, e.g., MDP.

Intuition:

Find (stochastic) shortest path on a discrete directed graph.

2.4.2 Converting to the State-Space Representation

It is useful to characterize the relationship between Formulation 2.4 and the original formulation of discrete feasible planning, Formulation 2.1. One benefit is that it immediately shows how to adapt the search methods of Section 2.2 to work for logic-based representations. It is also helpful to understand the relationships between the algorithmic complexities of the two representations.

Up to now, the notion of "state" has been only vaguely mentioned in the context of the STRIPS-like representation. Now consider making this more concrete. Suppose that every predicate has k arguments, and any instance could appear in each argument. This means that there are $|P|\,|I|^k$ complementary pairs, which corresponds to all of the ways to substitute instances into all arguments of all predicates. To express the state, a positive or negative literal must be selected from every complementary pair. For convenience, this selection can be encoded as a binary string by imposing a linear ordering on the instances and predicates.

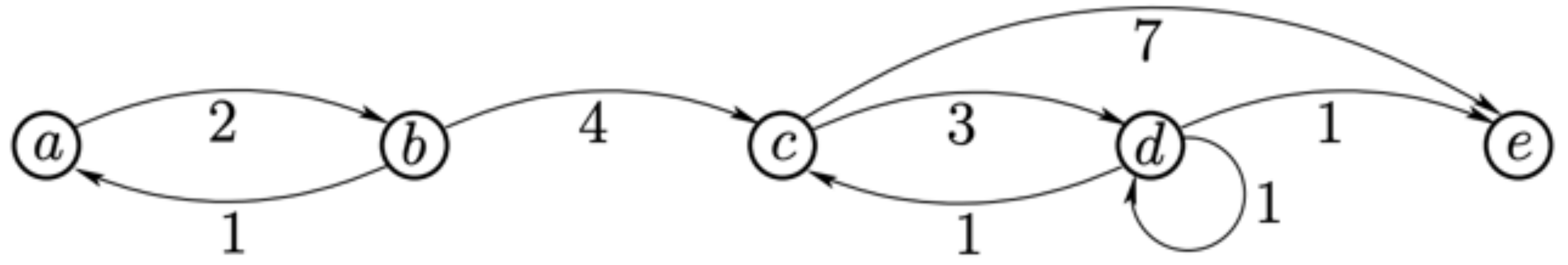


Figure 2.21: Another five-state discrete planning problem.

Planning Algorithms. Steven M. LaValle, 2006.

Basics of Planning: Summary

People in RL / optimal control and people in classic AI planning communities have different languages and serve for different purposes.

They are closely related while are suitable for different use cases.

Now — switch angle, how learning is helpful in these planning techniques.

Outline

Goals and Motivation

Basics of Planning

The Role of Learning in Planning

Planning Algorithms & Integration with Learning

Case Study: Mobile Manipulation

Takeaways

The Role of Learning in Planning

Where does learning come in

1, A (learned) state space

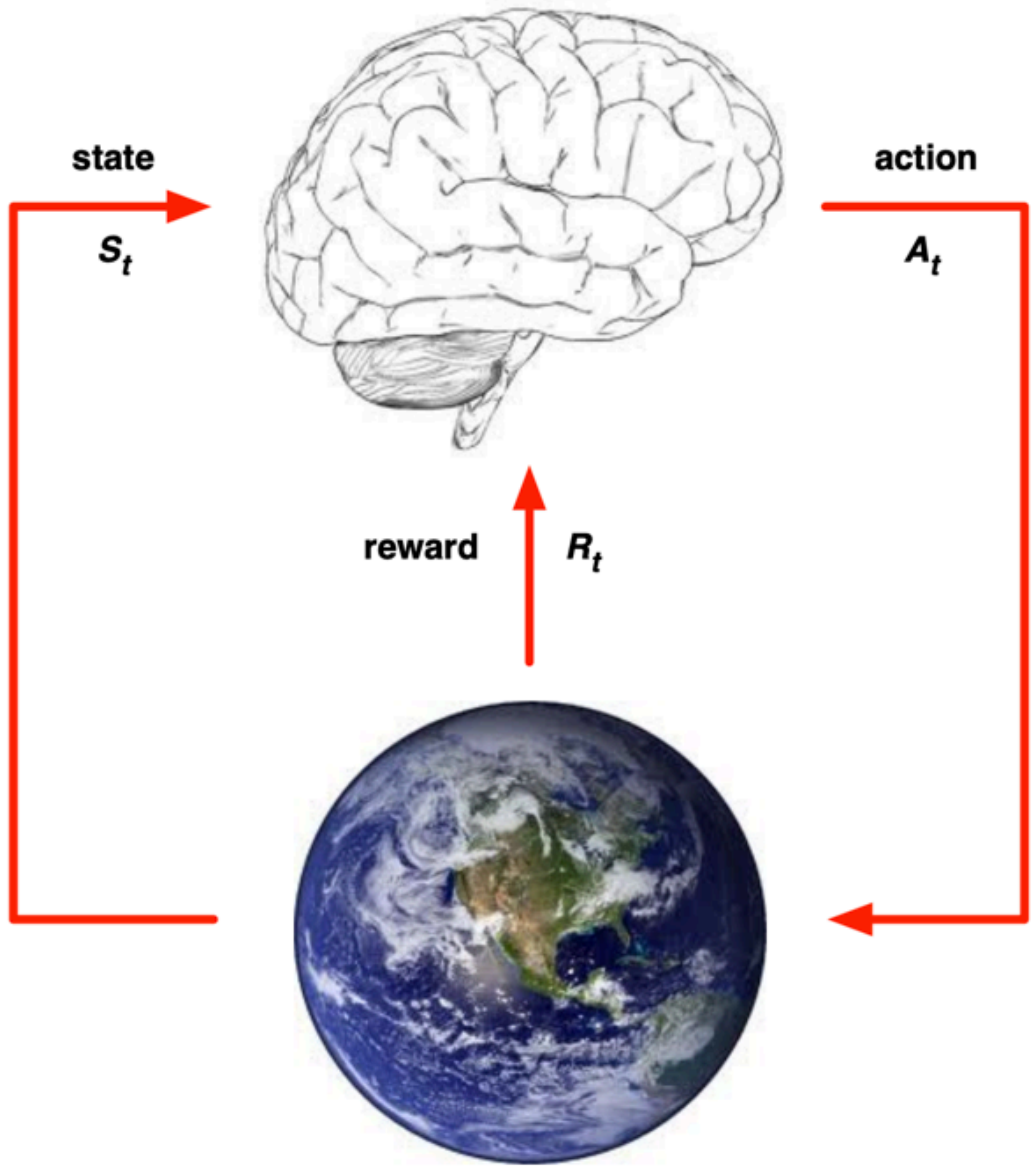
2, A (learned) transition model (world model)

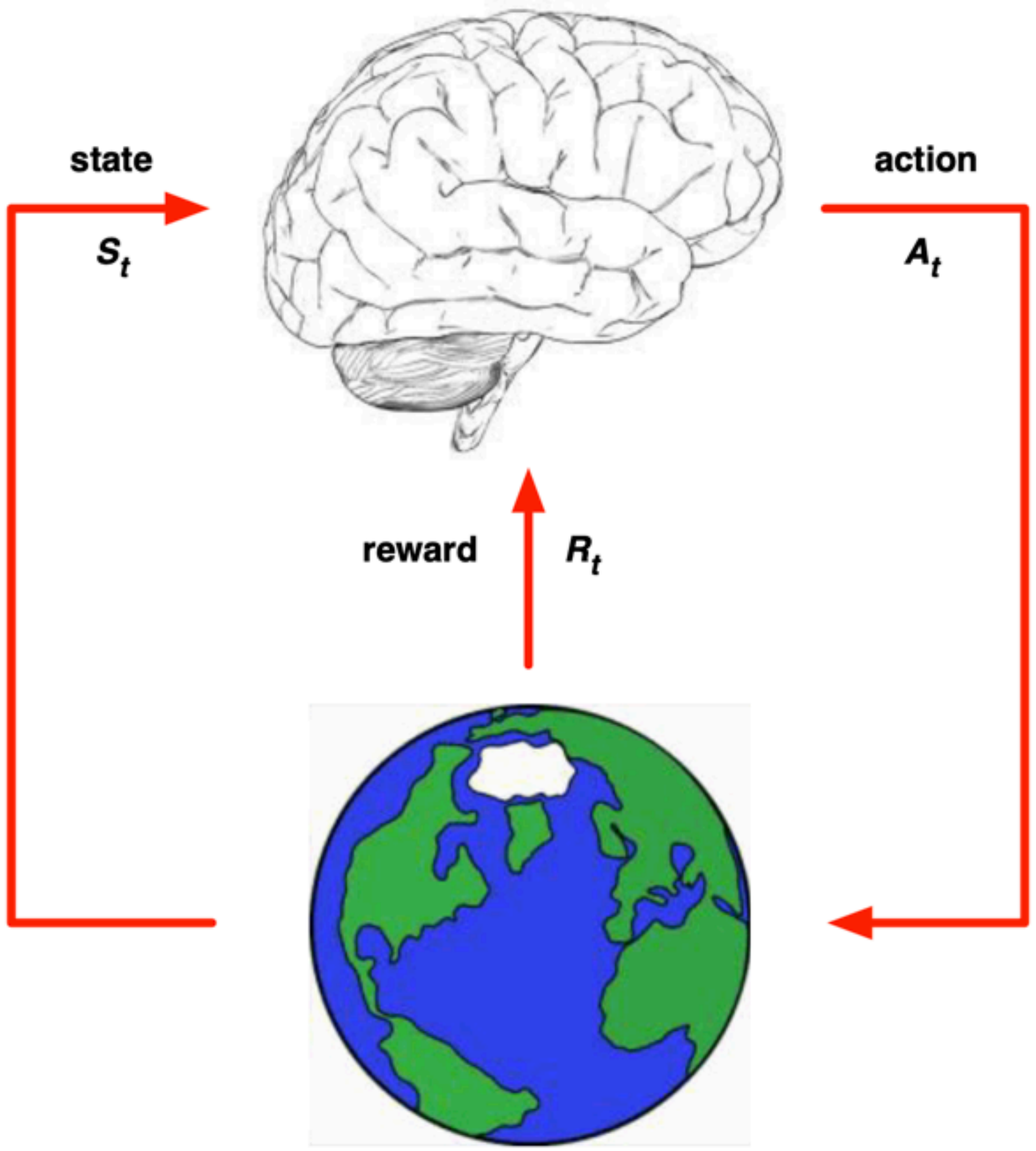
- $M = \langle P, R \rangle - P(s' \mid s, a)$ is transition dynamics, $R(s, a)$ is reward function
- Or deterministic case: $s', r = f(s, a)$

3, Planning algorithm & objective

- actions = Planner(state, goal)
- Objective: $$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

1. Learning World Models for Planning





Object-centric World Models

Figure 1: Mobile-manipulation robots operating in human-centric environments must know about, and be able to model, the world in terms of objects.

Learning the State of the World:
Object-based World Modeling for Mobile-Manipulation Robots

by

Lawson L.S. Wong

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

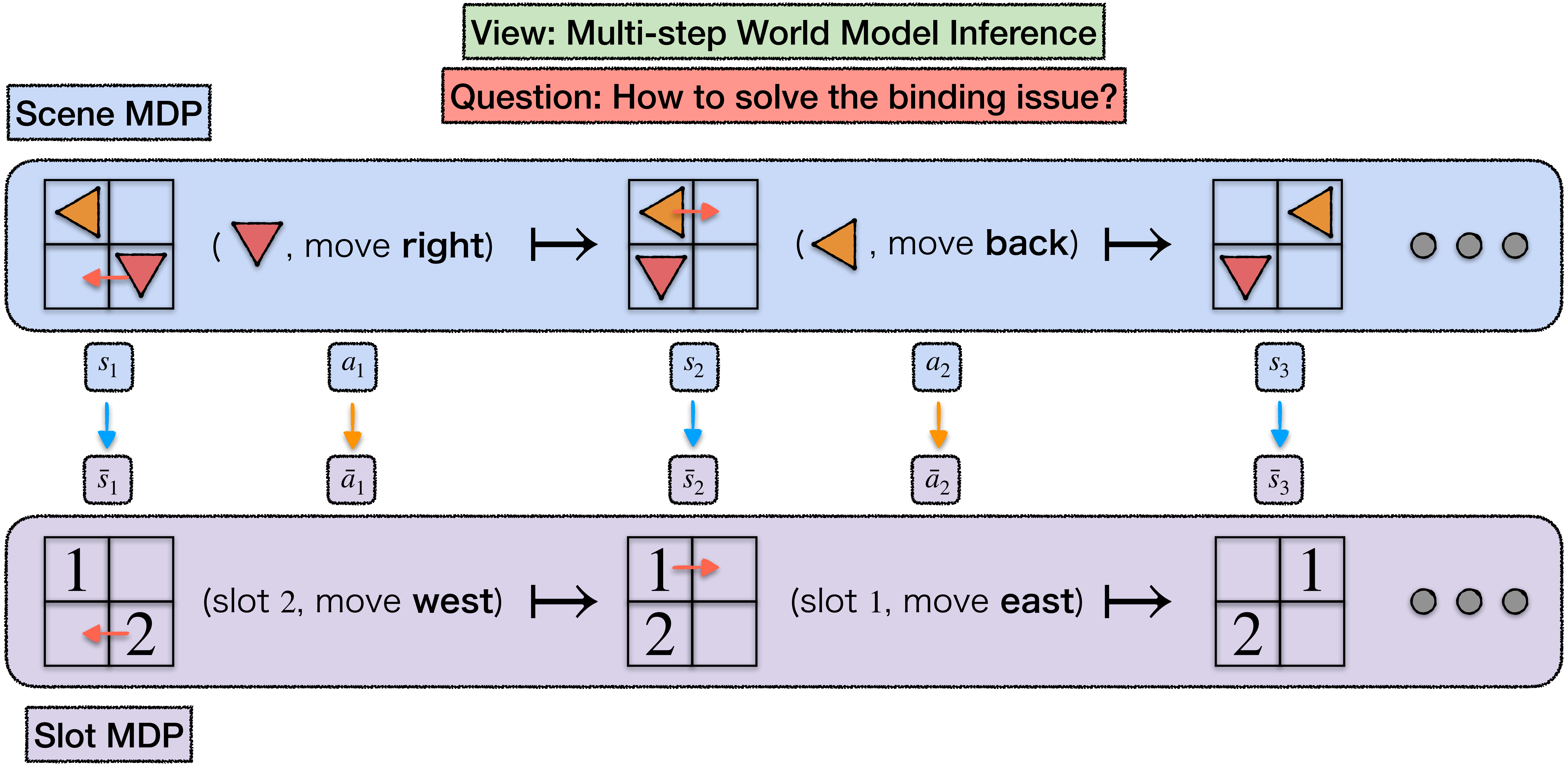
Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

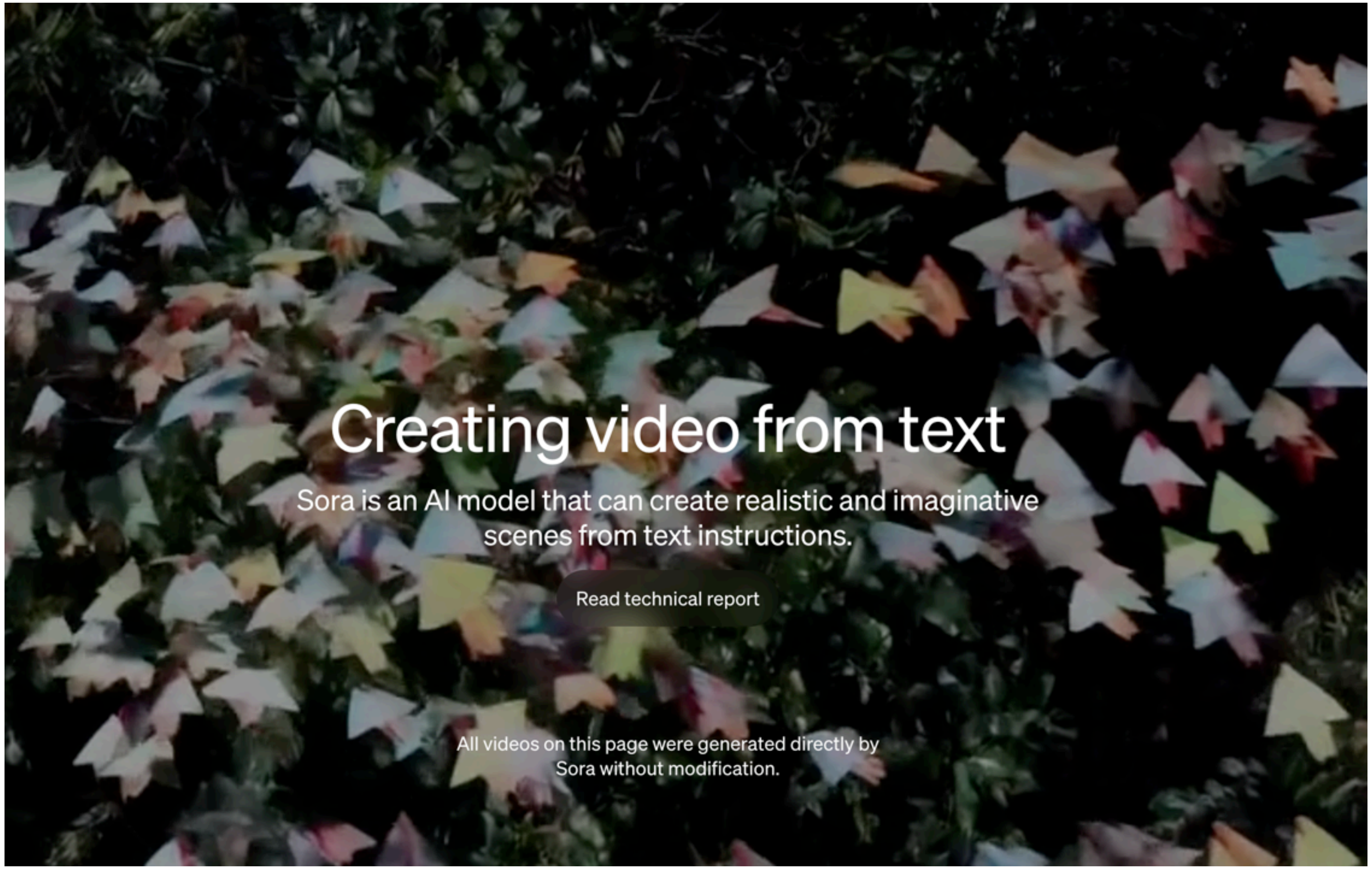
February 2016

Compositionality and Object-centric



Zhao et al. Toward Compositional Generalization in Object-Oriented World Modeling. ICML 2022.

Sora — video prediction



Video-language model (for planning)

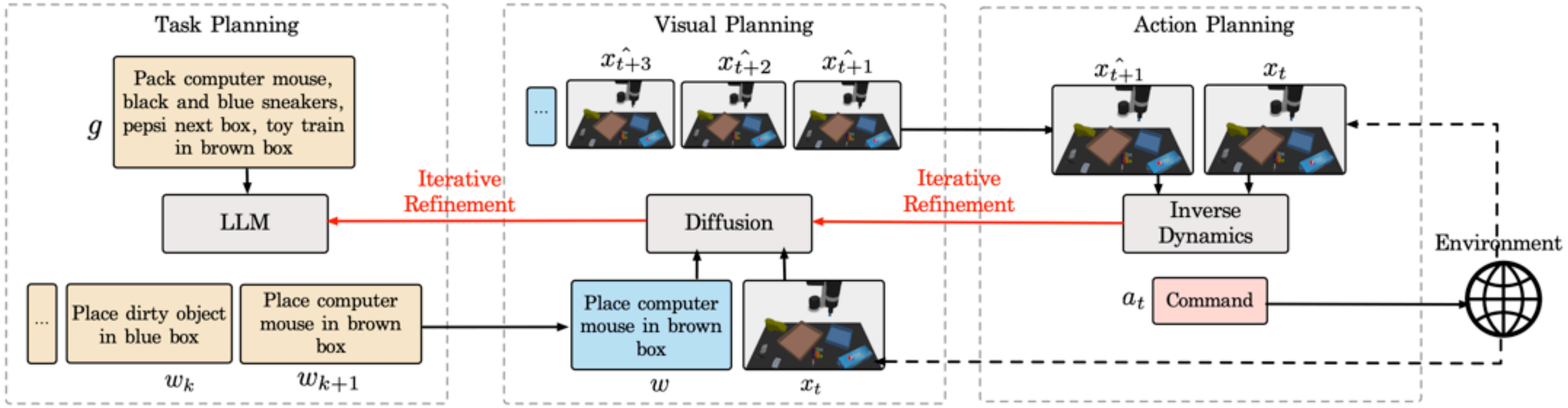
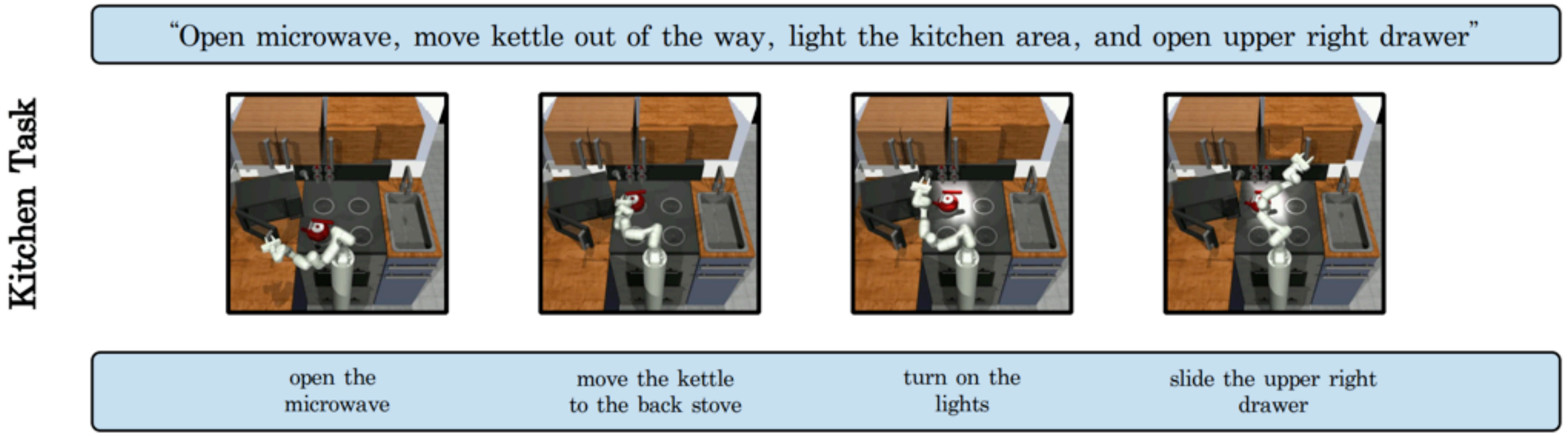


Figure 2: **Planning with** `HiP`**.** Given a language goal g and current observation x_t, LLM generates next subgoal w with feedback from a visual plausibility model. Then, Diffusion uses observation x_t and subgoal w to generate observation trajectory τ_x with feedback from an action feasibility model. Finally, action planning uses inverse dynamics to generate action a_t from current x_t and generated observation $\hat{x_{t+1}}$ (action planning).

Video prediction for next frame

With/without actions



2. Learning Representations for Planning

We want to learn a compact and abstract representation of the world

Original transition dynamics $p(s' \mid s, a)$ or $s' = f(s, a)$ may be too hard to learn

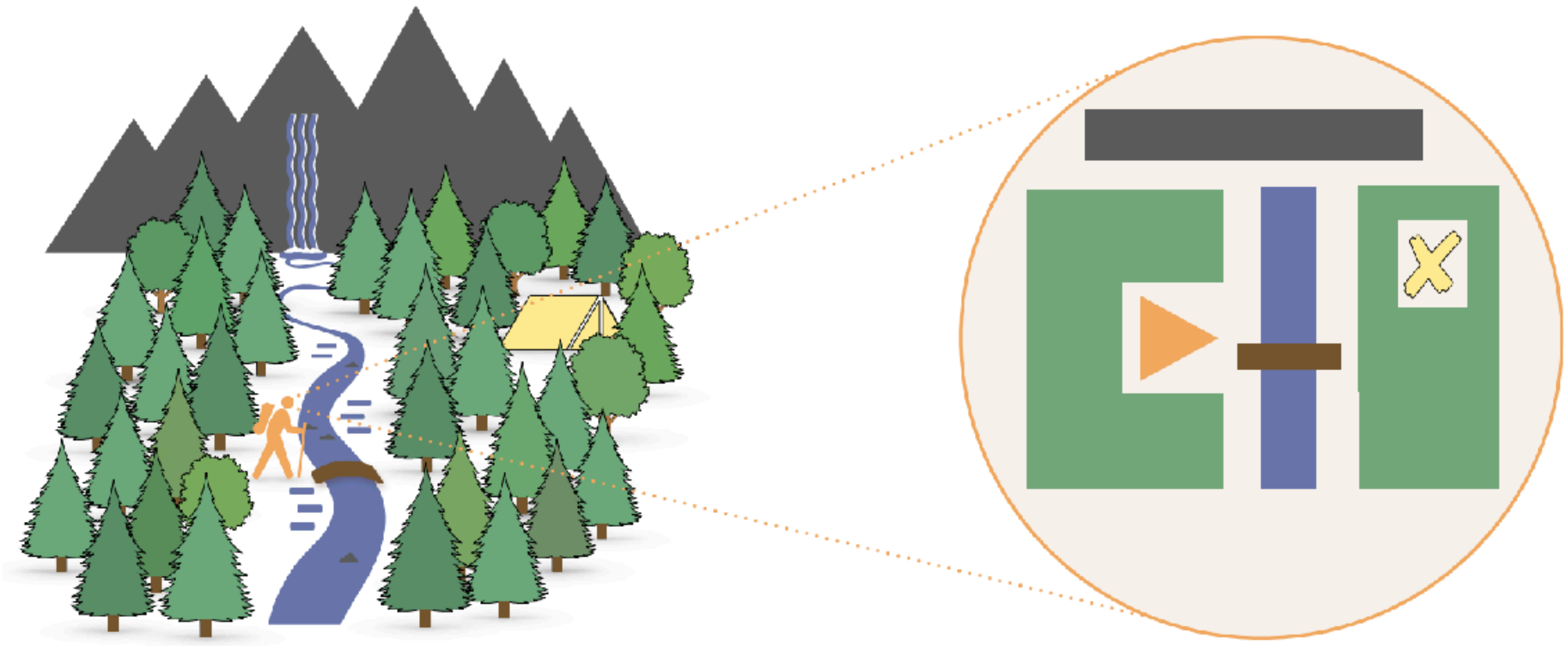


Figure 1.1: The process of abstraction.

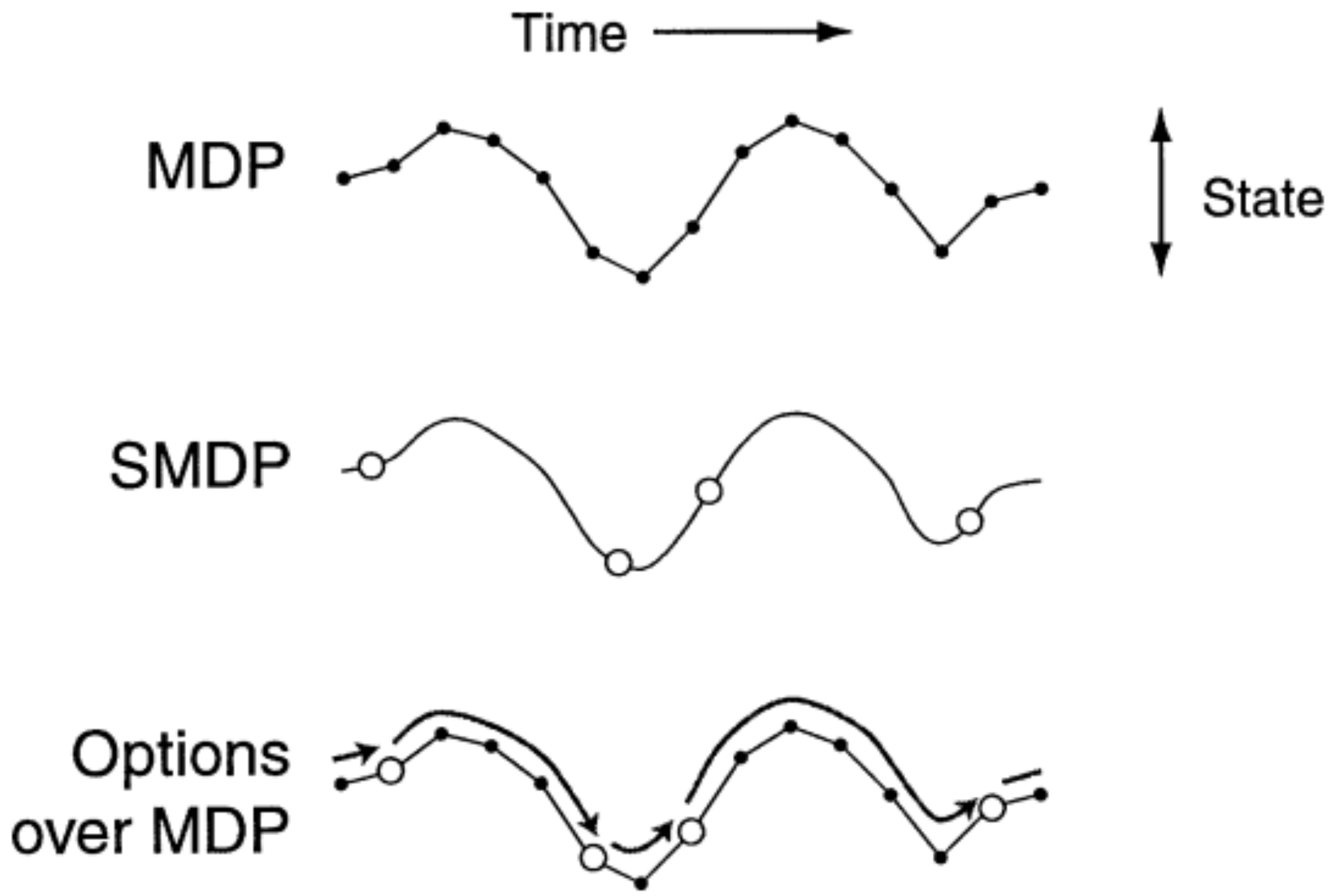
A Theory of Abstraction in Reinforcement Learning. David Abel, PhD Thesis 2020.

Action Abstraction: Options framework

Options:

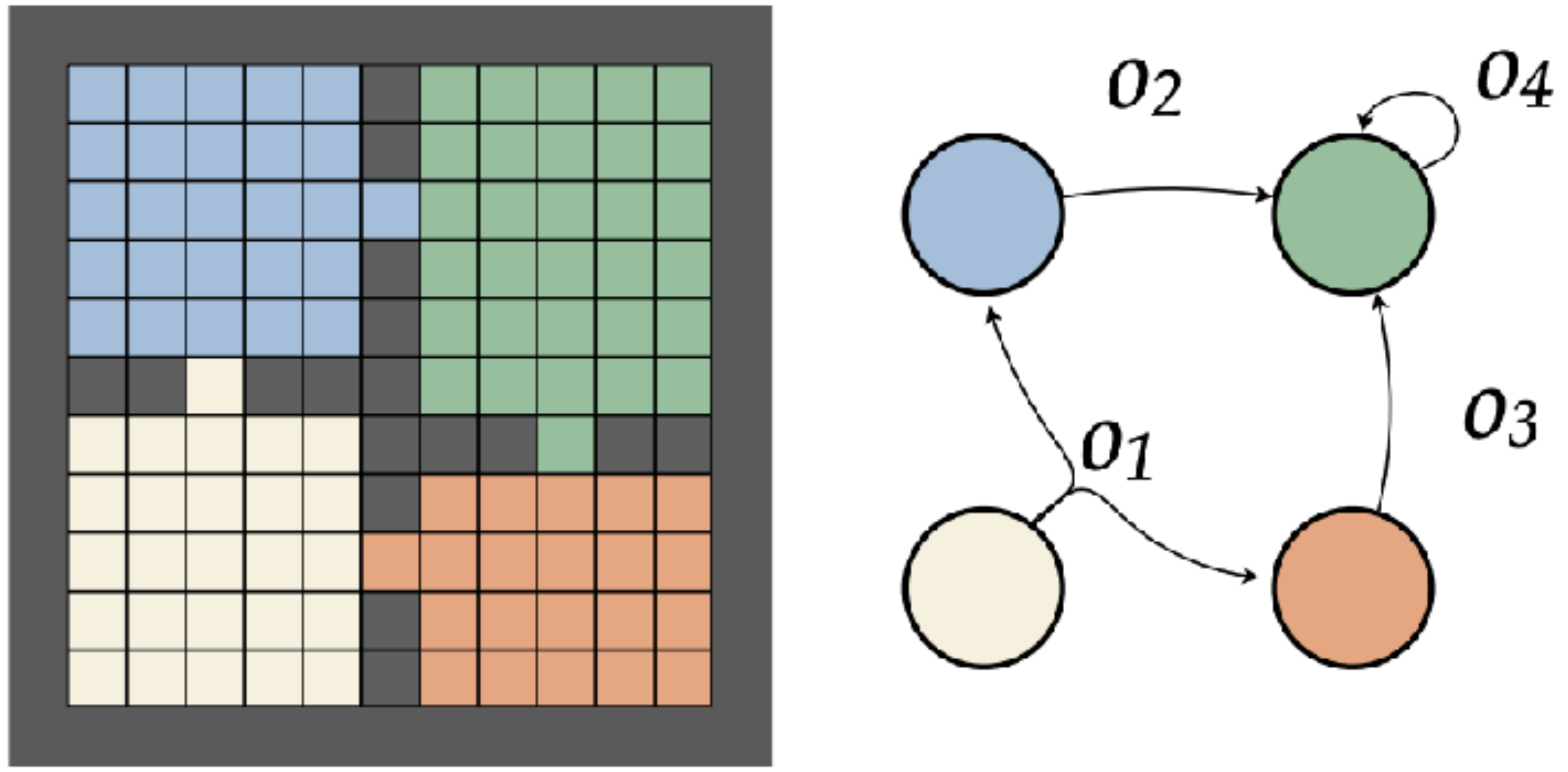
Temporally extended actions

Developed in *Semi-MDP*

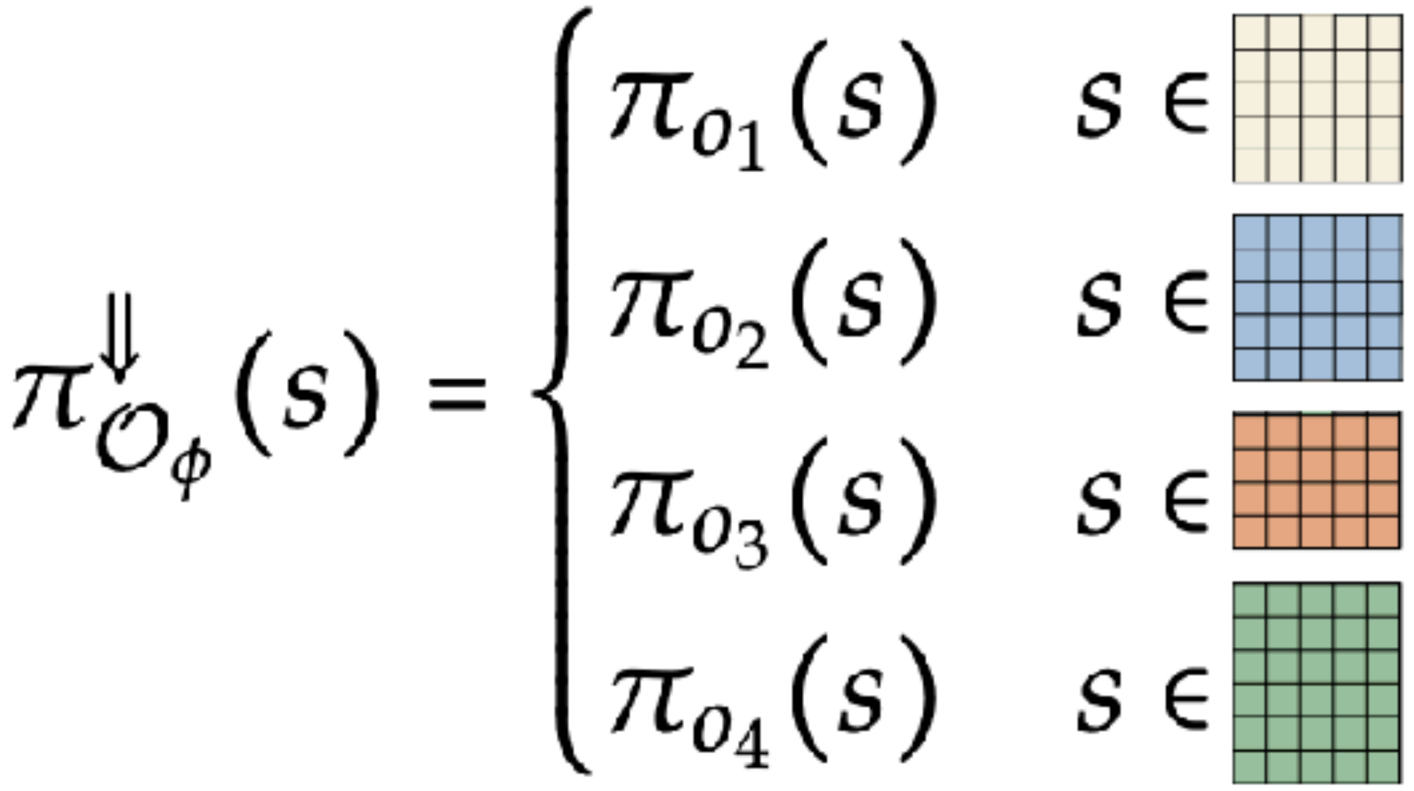


Reinforcement Learning: An Introduction. Andrew Barto and Richard S. Sutton 2018.

State Abstraction: via State Partition



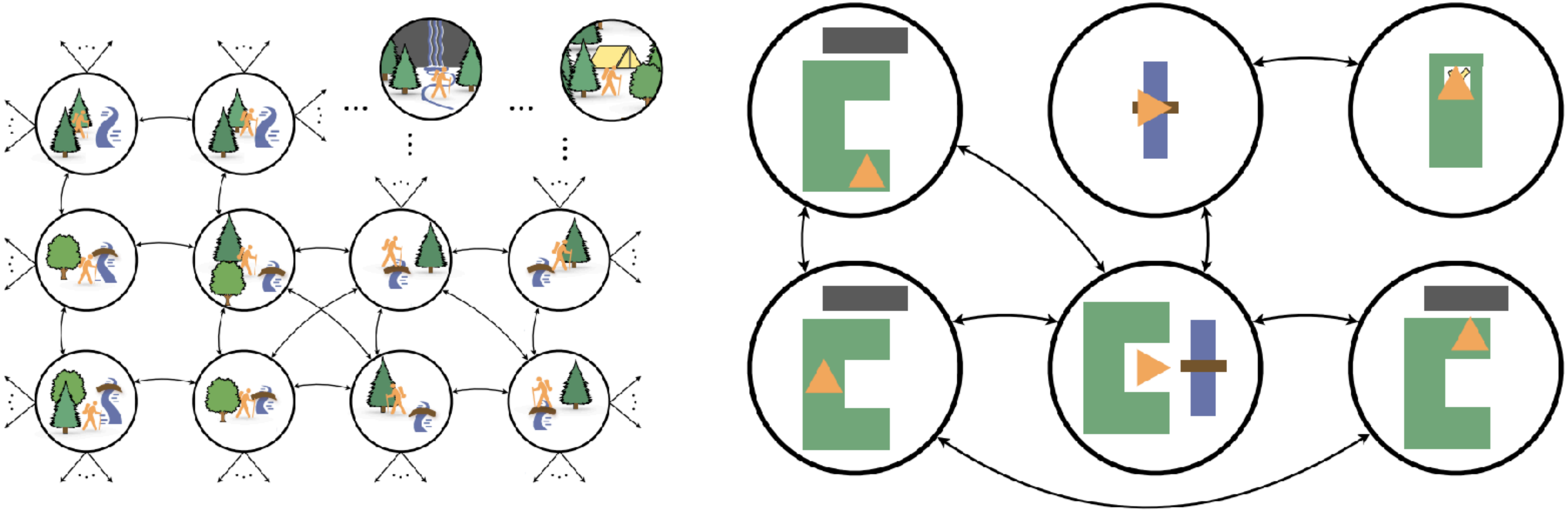
(a) Assignment of options to each s_ϕ via $\pi_{\mathcal{O}_\phi}$.



(b) Construction of $\pi^{\Downarrow}_{\mathcal{O}_\phi}$.

A Theory of Abstraction in Reinforcement Learning. David Abel, PhD Thesis 2020.

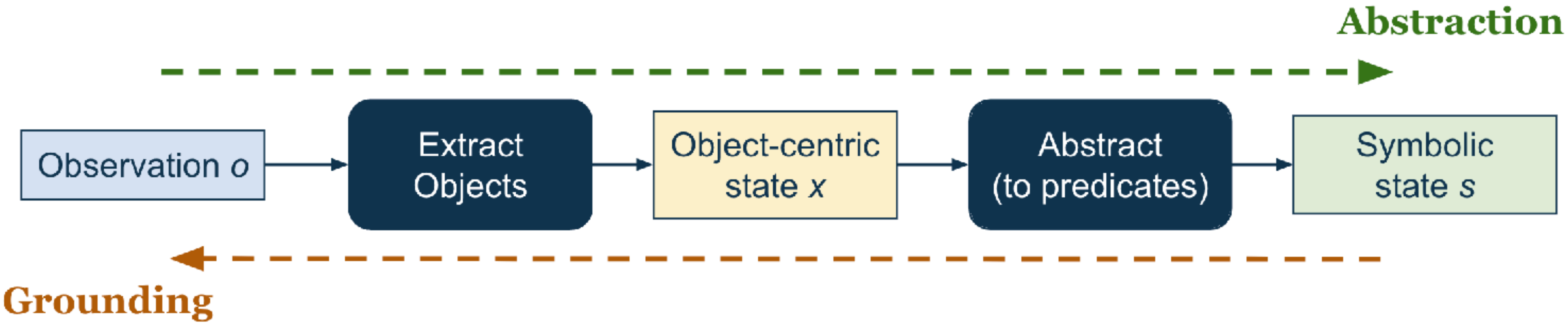
Planning in the abstracted model

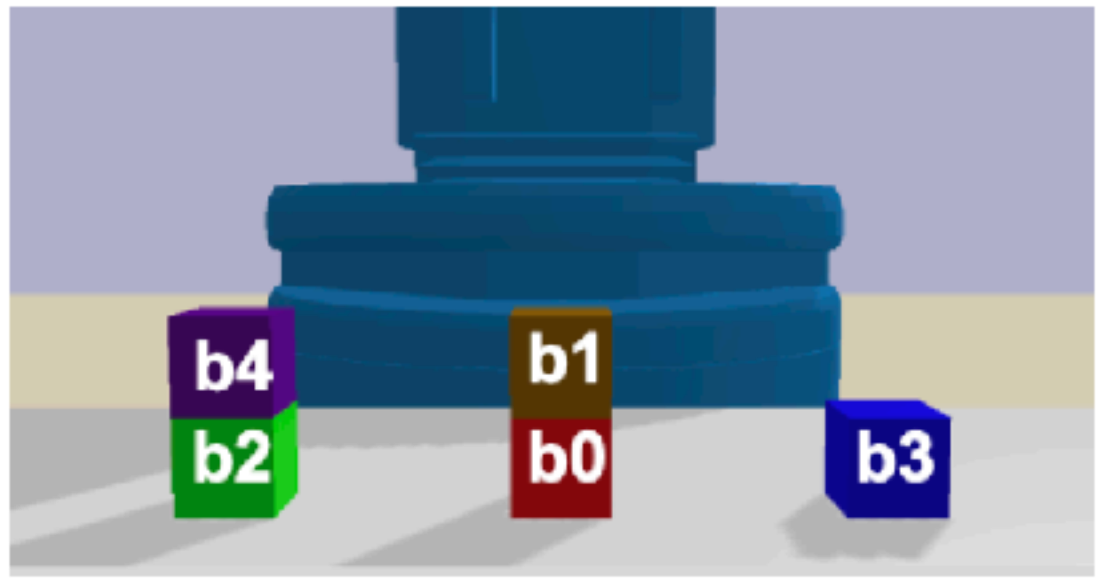


(a) Reasoning in the environment. (b) Reasoning in the abstract.

A Theory of Abstraction in Reinforcement Learning. David Abel, PhD Thesis 2020.

Object-centric State Representations





	x	y	z	size
rob	0.63	0.11	0.94	0.5
b0	0.74	0.11	0.00	0.1
b1	0.75	0.10	0.20	0.1
b2	0.50	0.11	0.00	0.1
b3	0.99	0.12	0.00	0.1
b4	0.51	0.11	0.20	0.1

```
OnTable(b2), On(b4, b2)
OnTable(b0), On(b1, b0)
OnTable(b3)
```

Abstract Action: Operators/Skills

Used in STRIPS and PDDL:

```
Grasp-From-On-Top
    Parameters:      [?target: obj, ?surface: obj]
    Preconditions:   [HandEmpty,
                      Reachable(?target),
                      OnTop(?target, ?surface)]
    Add Effects:     [Holding(?target)]
    Delete Effects:  [HandEmpty,
                      Reachable(?target),
                      OnTop(?target, ?surface)]
    Skill:           Grasp(?target, [Δx, Δy, Δz])
```

Figure 5: Example operator for grasping from atop a surface. The operator has two arguments (both of type 'obj'): a target object to pick up, and a surface from which to pick this object. The operator's associated skill is parameterized by the discrete target object, as well as three continuous parameters that correspond to a cartesian position in the object's coordinate frame at which the robot should attempt to grasp the object..

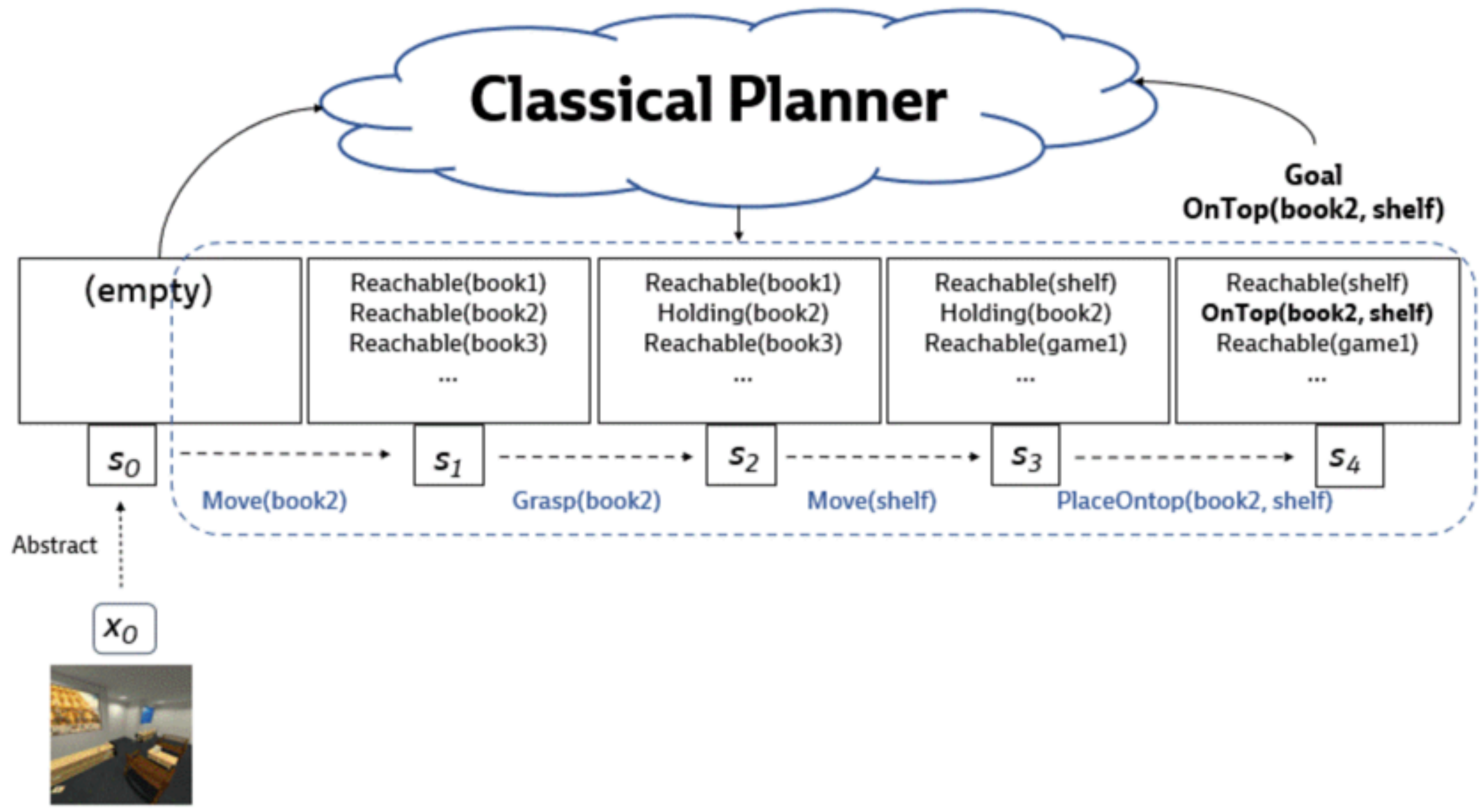


Figure 6: Animated visualization of high-level planning with the provided abstractions. We first abstract our initial low-level state (x0) into an initial high-level state (s0), then use an off-the-shelf AI planning system to come up with a sequence of ground operators that achieve the goal conditions.

Bilevel Planning for Robots: An Illustrated Introduction. Kumar et al. Blog.

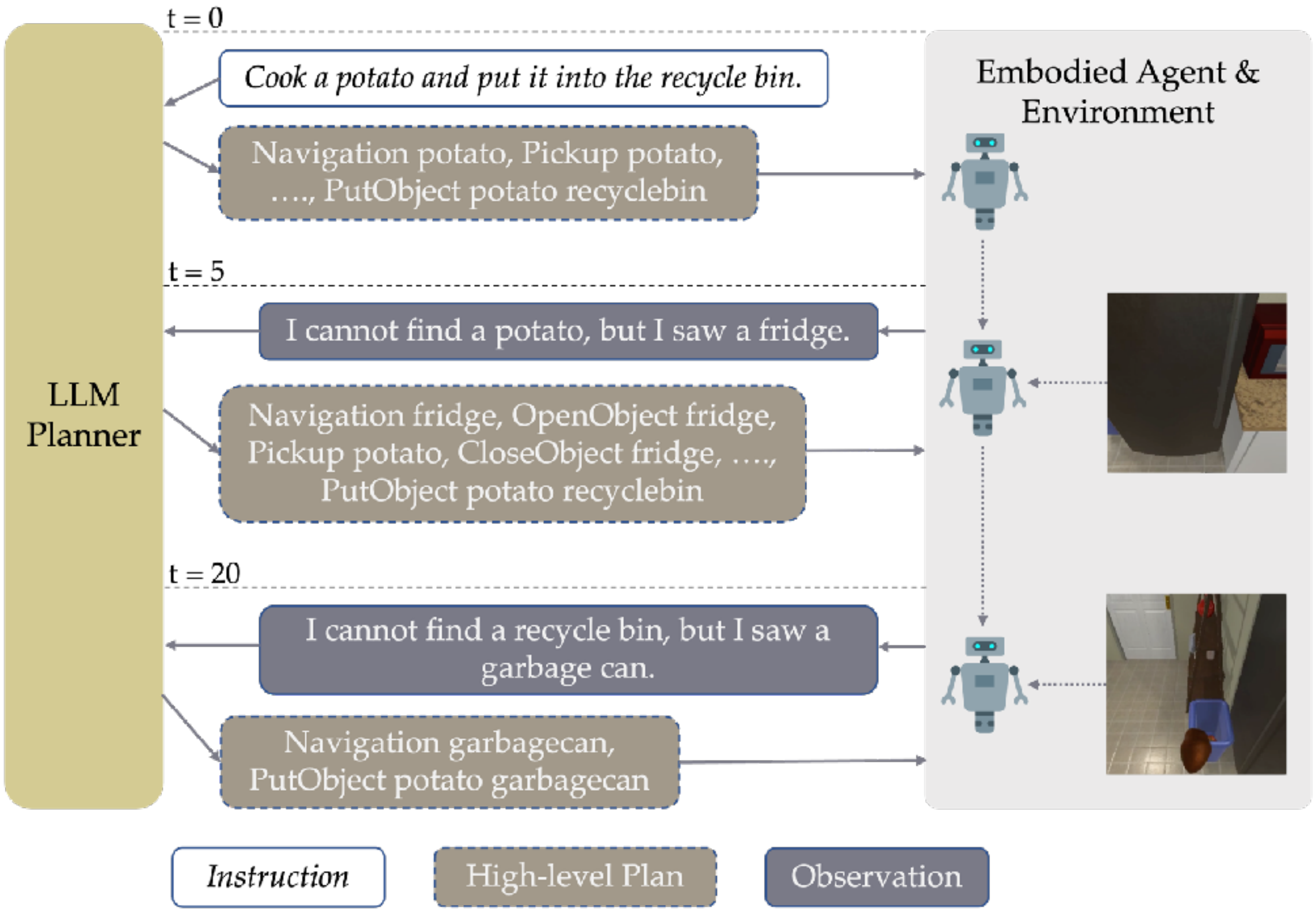
3. Learning Planning Computation

Normally we have an algorithm to do planning computation:

actions = Planner(state, goal)

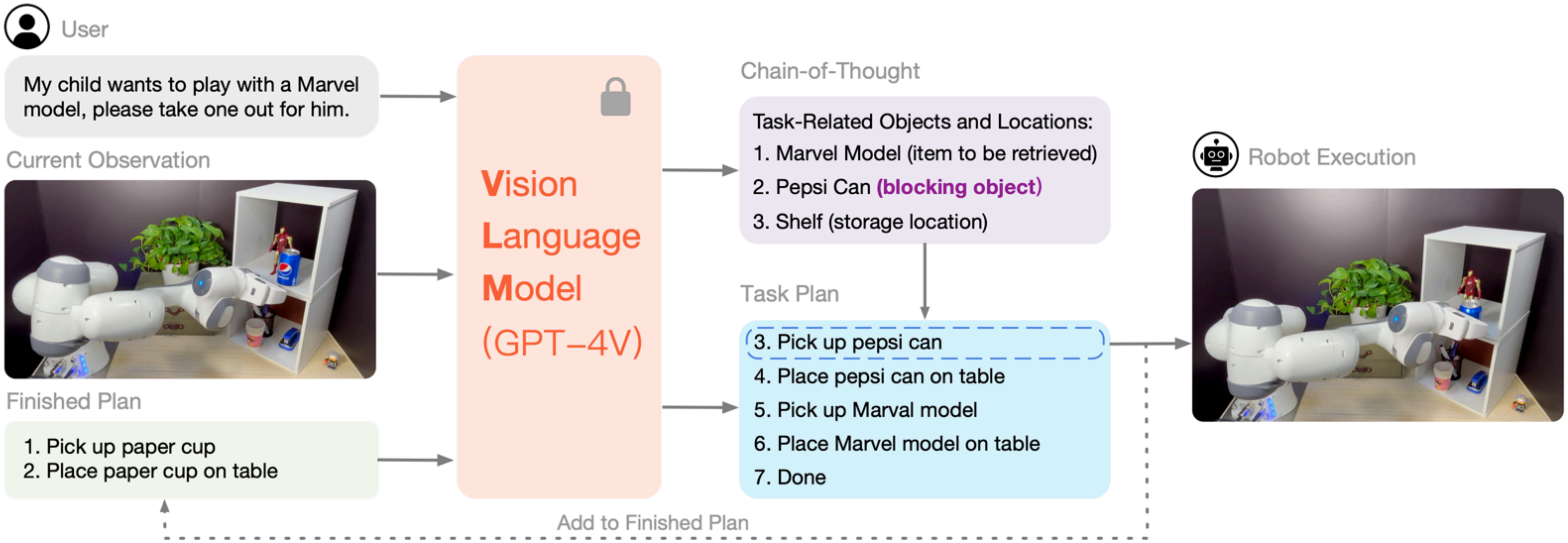
This can also be done approximately via a neural network, e.g., LLM with Transformer:

actions = LLM(state, goal)



Song et al. LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models.

Using VLMs to Plan



Hu et al. Look Before You Leap: Unveiling the Power of GPT-4V in Robotic Vision-Language Planning. arXiv 2023.

**This paper has been using teleop for real-robot demo*

Section Summary

Learning is helpful for various places for planning!

How can we really use them in planning?

Next:

Go over several types of planning algorithms

Analyze why and how learning is helpful

Outline

Goals and Motivation

Basics of Planning

The Role of Learning in Planning

Planning Algorithms & Integration with Learning

Case Study: Mobile Manipulation

Takeaways

Planning Algorithms & Integration with Learning

Overview of planning algorithms

1. High-level / Discrete Space Planning

2. Low-level / Continuous Space Planning

3. Planning in Hybrid Space

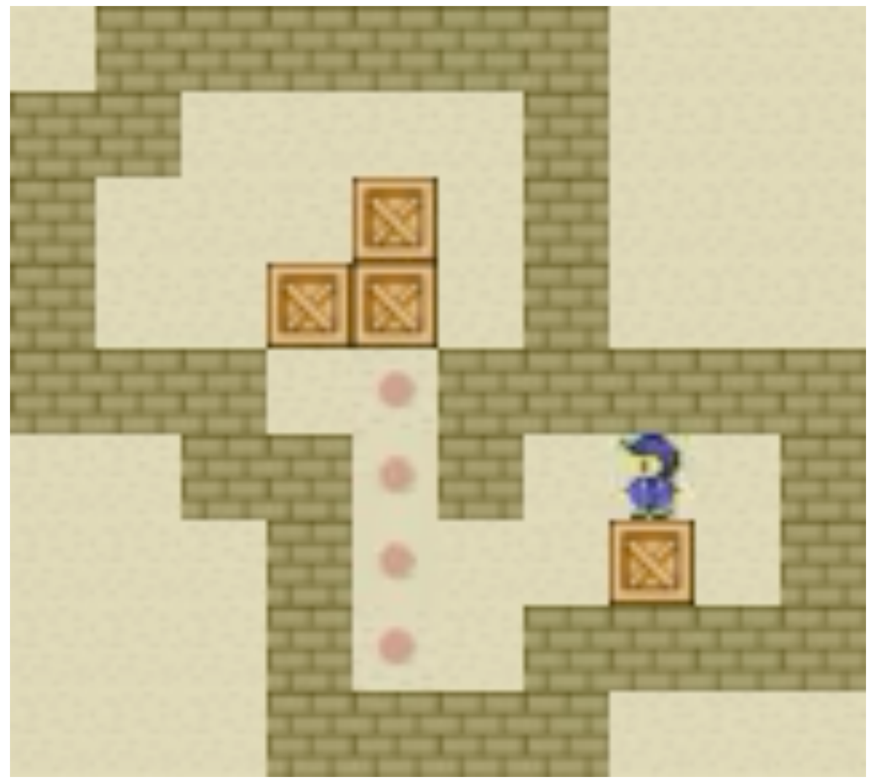
Goals:

Go over several types of planning algorithms

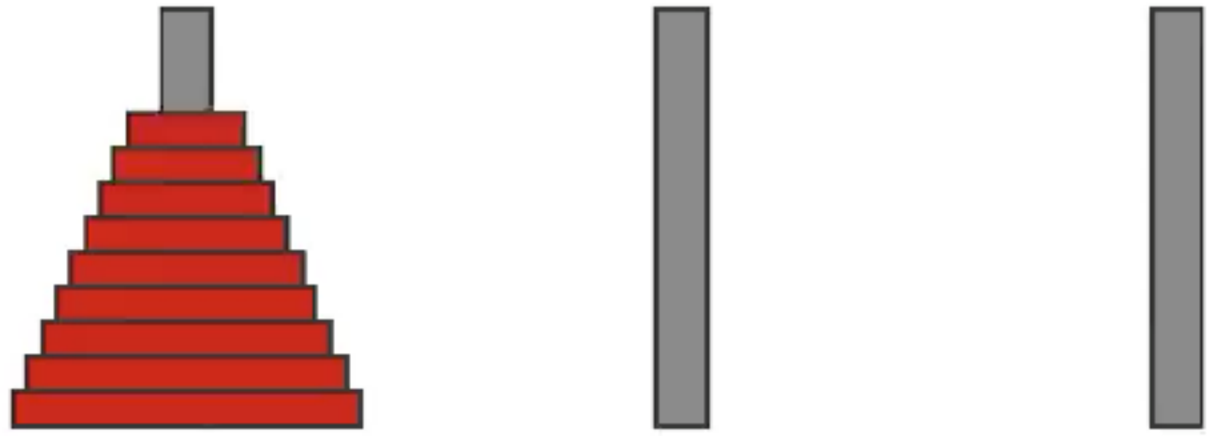
Analyze why and how learning is helpful

1. High-level / Discrete Space Planning

Blocks World
Plan length: 28
Planning time: 0.12 s



Sokoban
Plan length: 167
Planning time: 0.25 s



Hanoi
Plan length: 579
Planning time: 0.22 s

[Credit: Tom Silver]

Example: MCTS

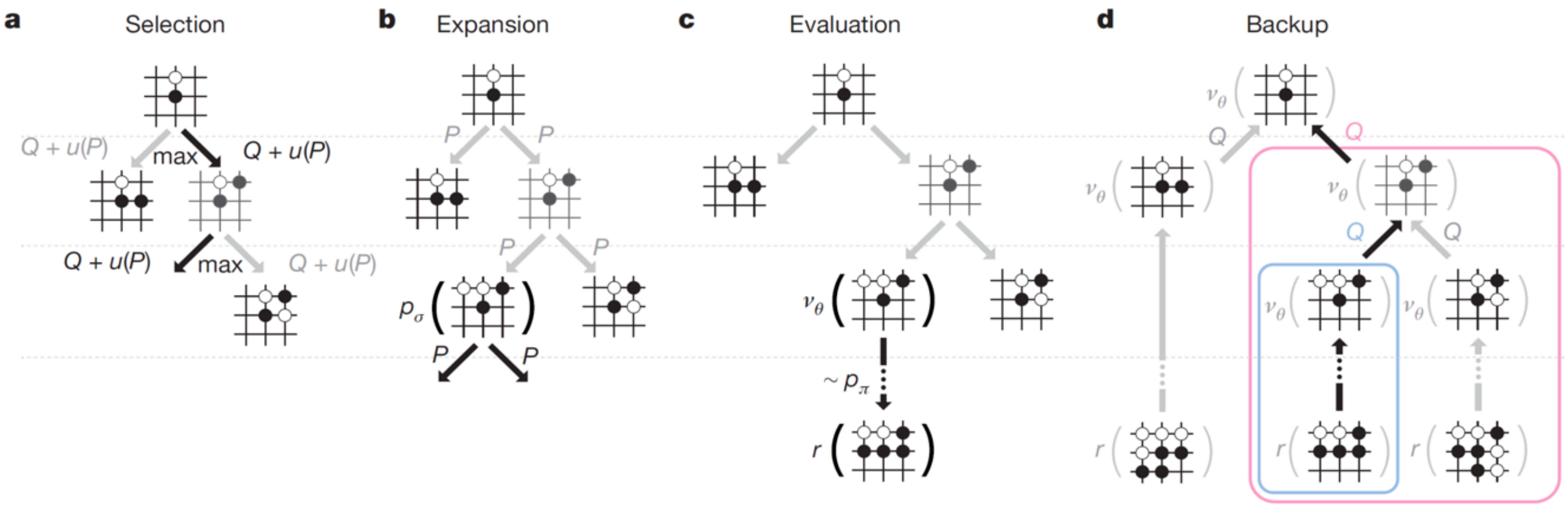
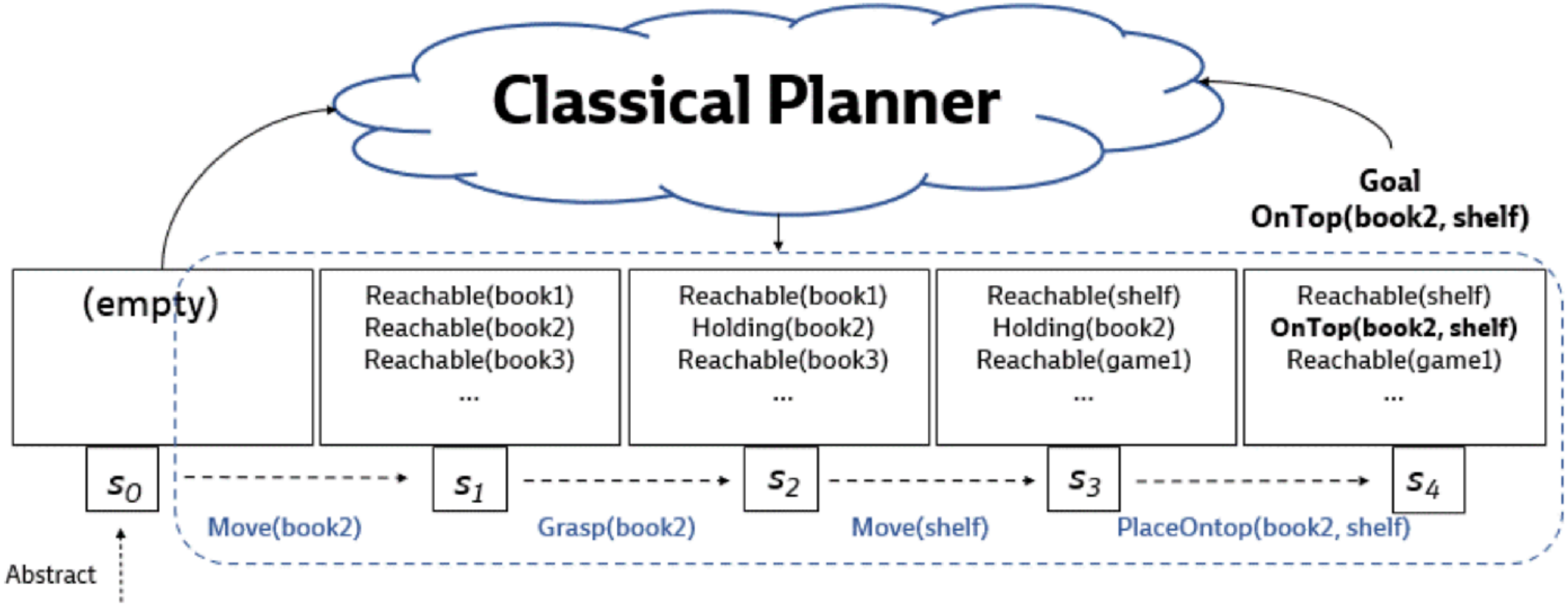


Figure 3 | Monte Carlo tree search in AlphaGo. a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus a bonus $u(P)$ that depends on a stored prior probability P for that edge. **b**, The leaf node may be expanded; the new node is processed once by the policy network p_σ and the output probabilities are stored as prior probabilities P for each action. **c**, At the end of a simulation, the leaf node is evaluated in two ways: using the value network v_θ; and by running a rollout to the end of the game with the fast rollout policy p_π, then computing the winner with function r. **d**, Action values Q are updated to track the mean value of all evaluations $r(\cdot)$ and $v_\theta(\cdot)$ in the subtree below that action.

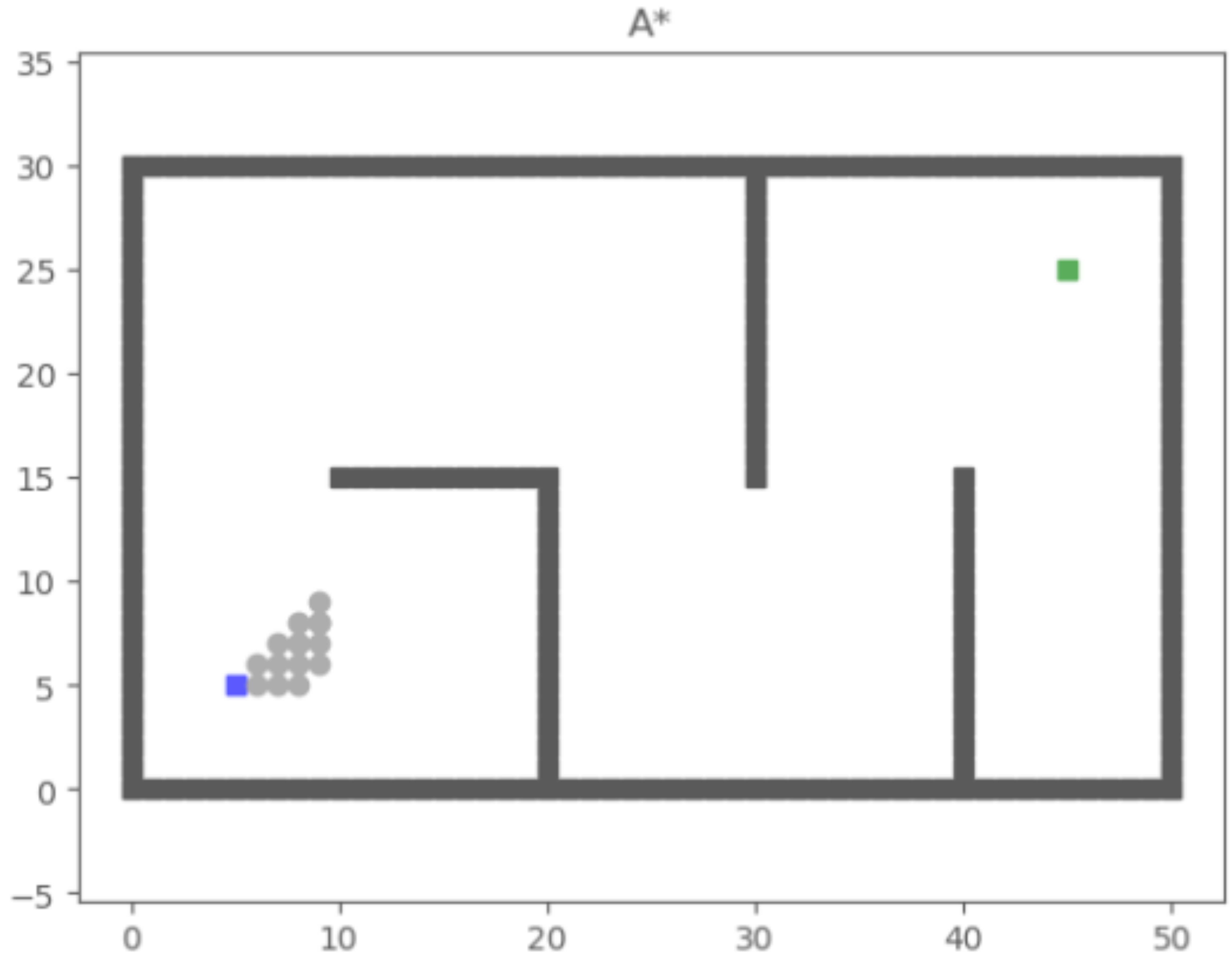
Related algorithms: AlphaGo, AlphaZero, MuZero — from Google DeepMind.

Example: PDDL Planning



Bilevel Planning for Robots: An Illustrated Introduction. Kumar et al. Blog.

Example: A*



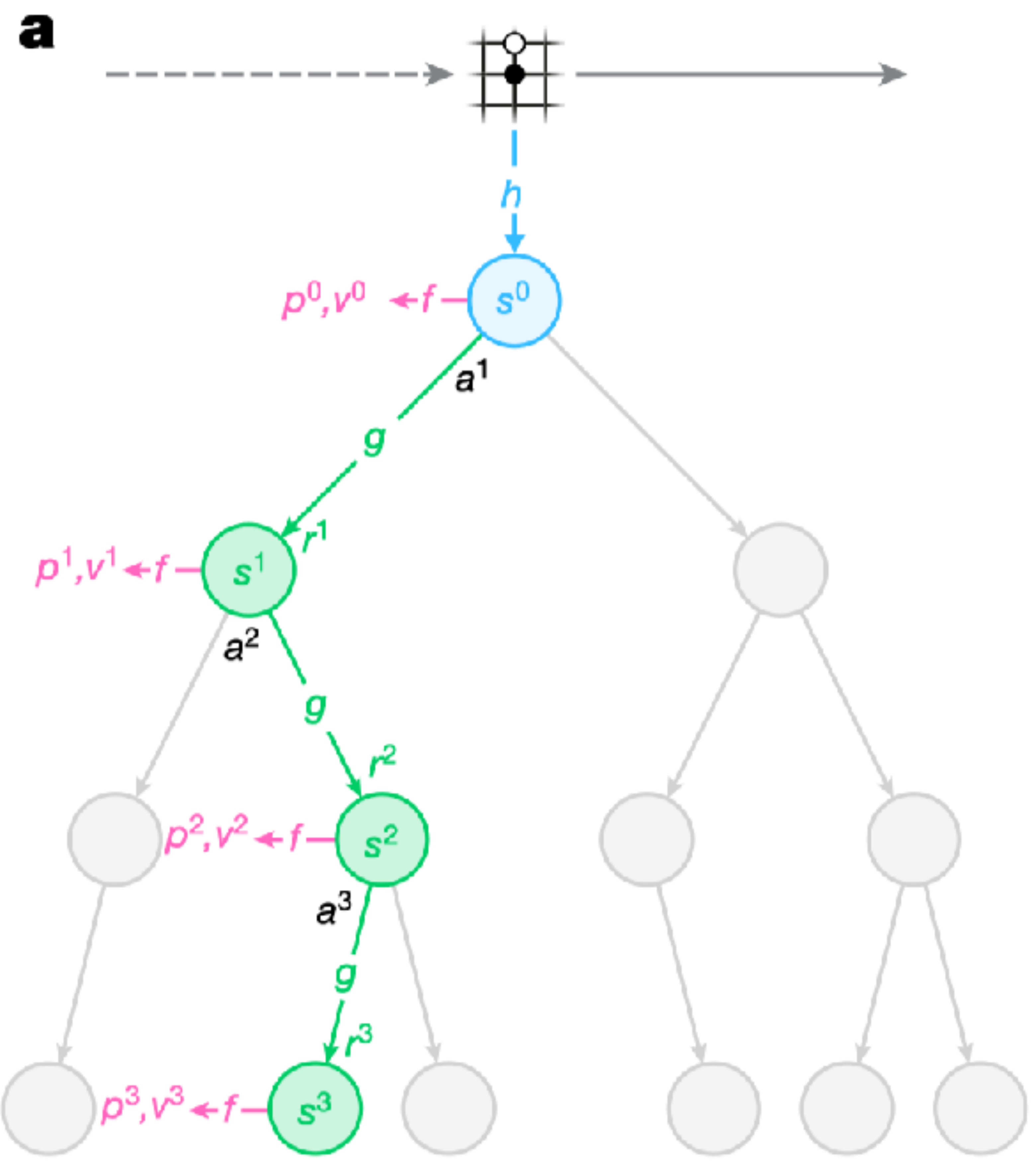
[https://github.com/zhm-real/PathPlanning]

Learning-based: MuZero

1, Representation: learned state encoder

2, Transition Model: learned MLP

3, Planning algorithm: MCTS



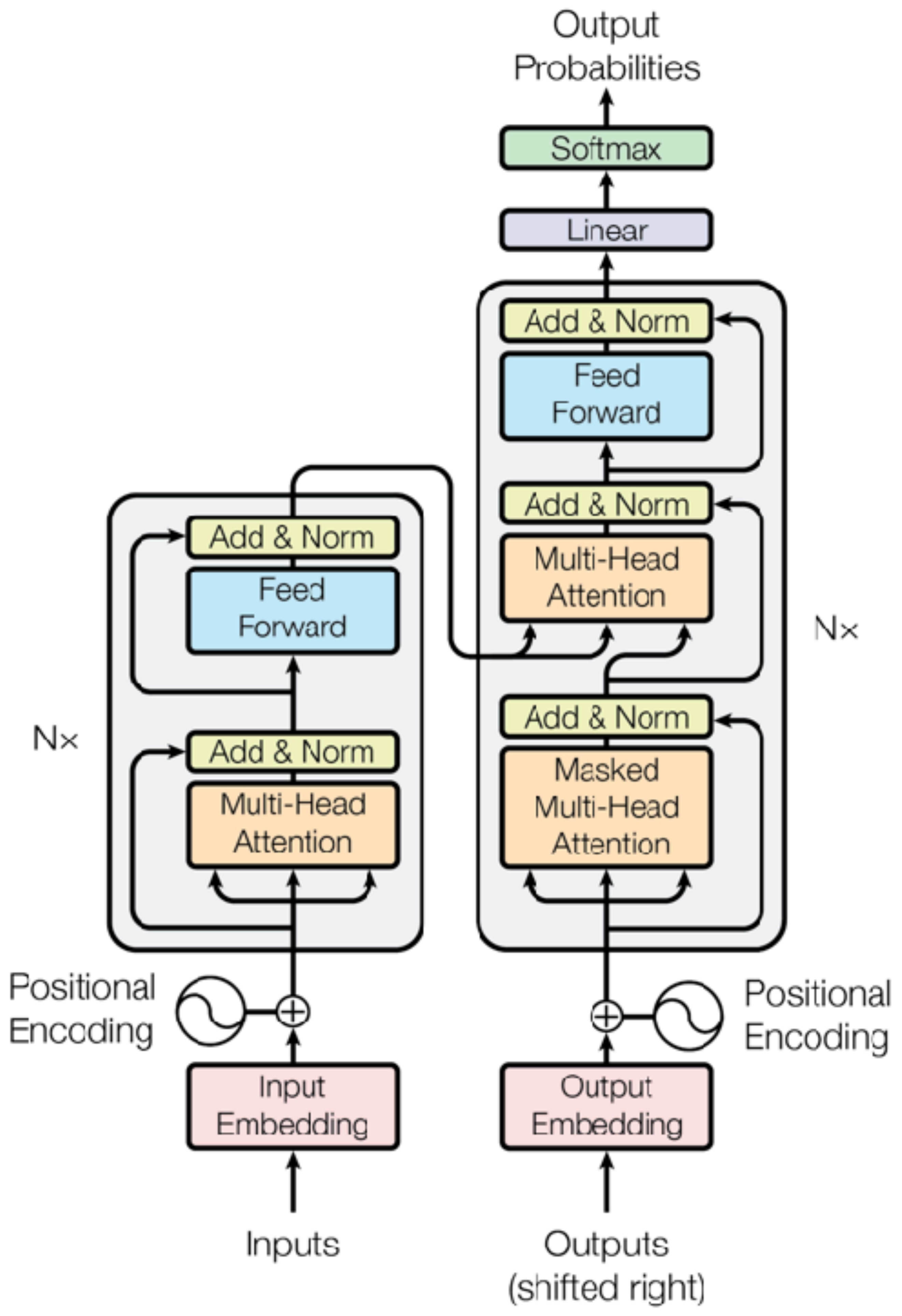
[Credit: MuZero, DeepMind]

Learning-based: LLM/VLM planning

1, Representation: language/vision tokenizer

2, Transition Model: learned Transformer

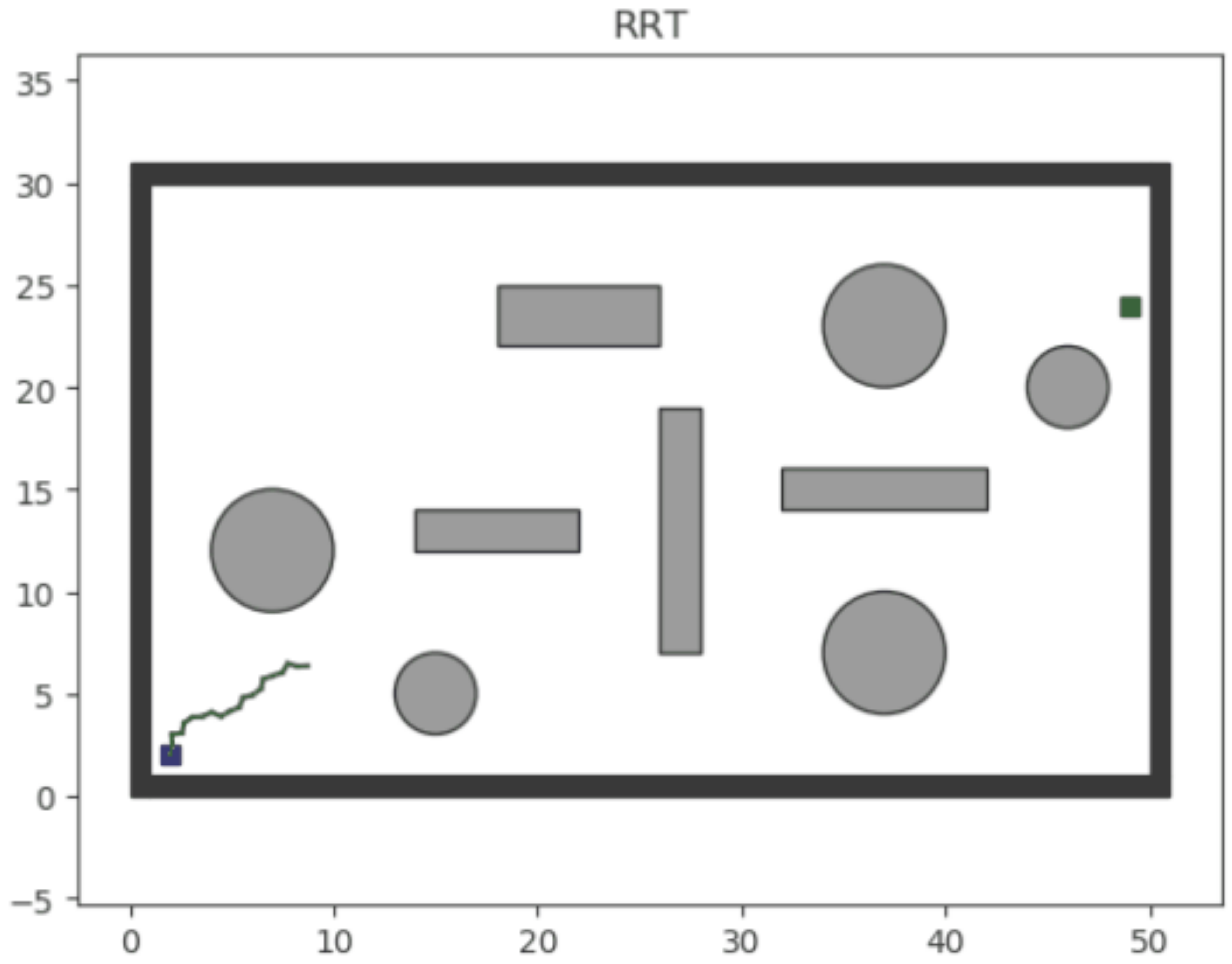
3, Planning algorithm: learned Transformer



Vaswani* et al. Attention is all you need. NIPS 2017.

2. Low-level / Continuous Space Planning

Example: RRT (rapidly-exploring random trees)

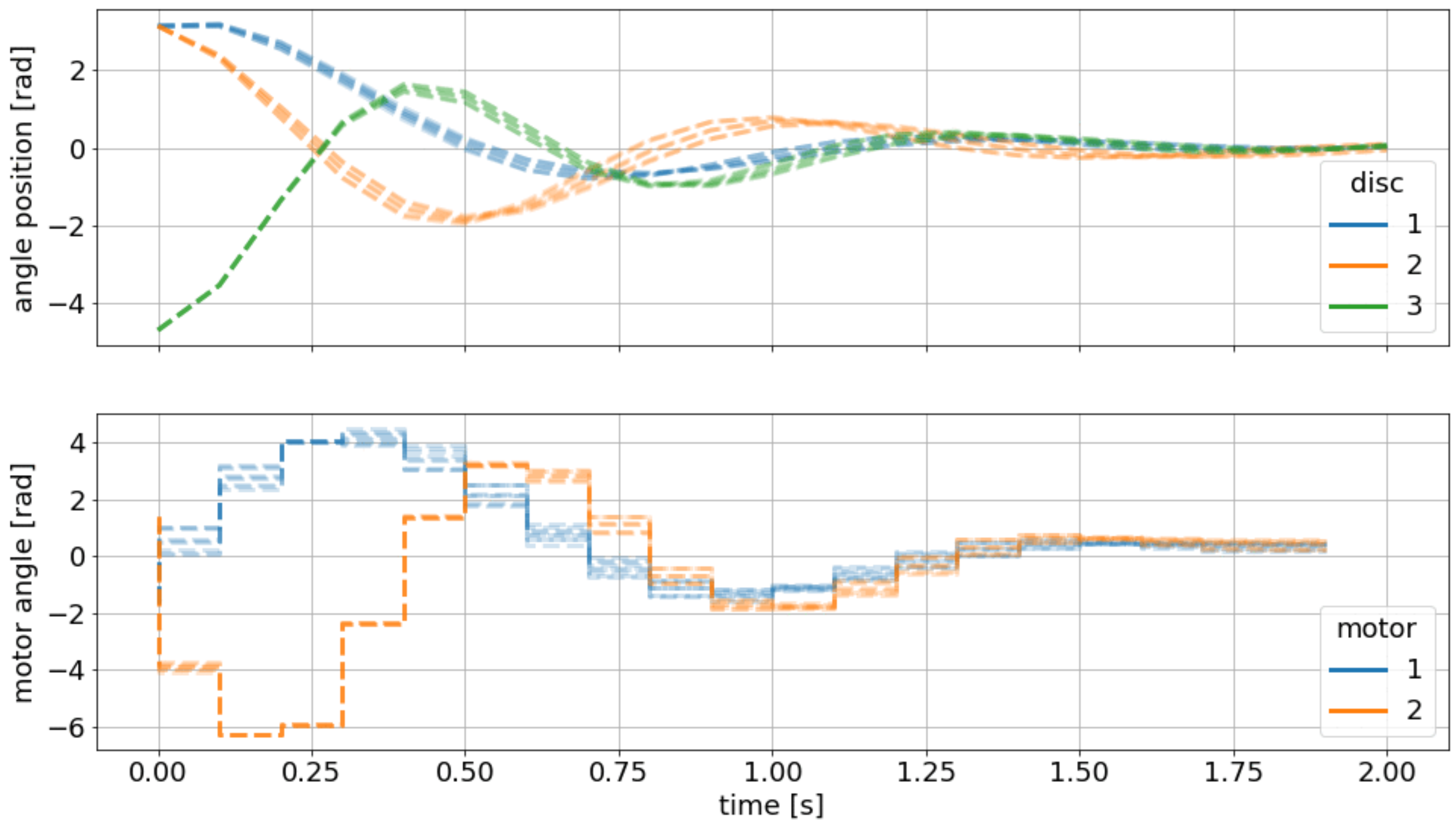


[https://github.com/zhm-real/PathPlanning]

Example: MPC (Model Predictive Control)

Idea:
- Predict a few states into the future
- Follow the actions from the better path

Another name:
Receding horizon control



[https://www.do-mpc.com/]

MPC — example task

Example Task:

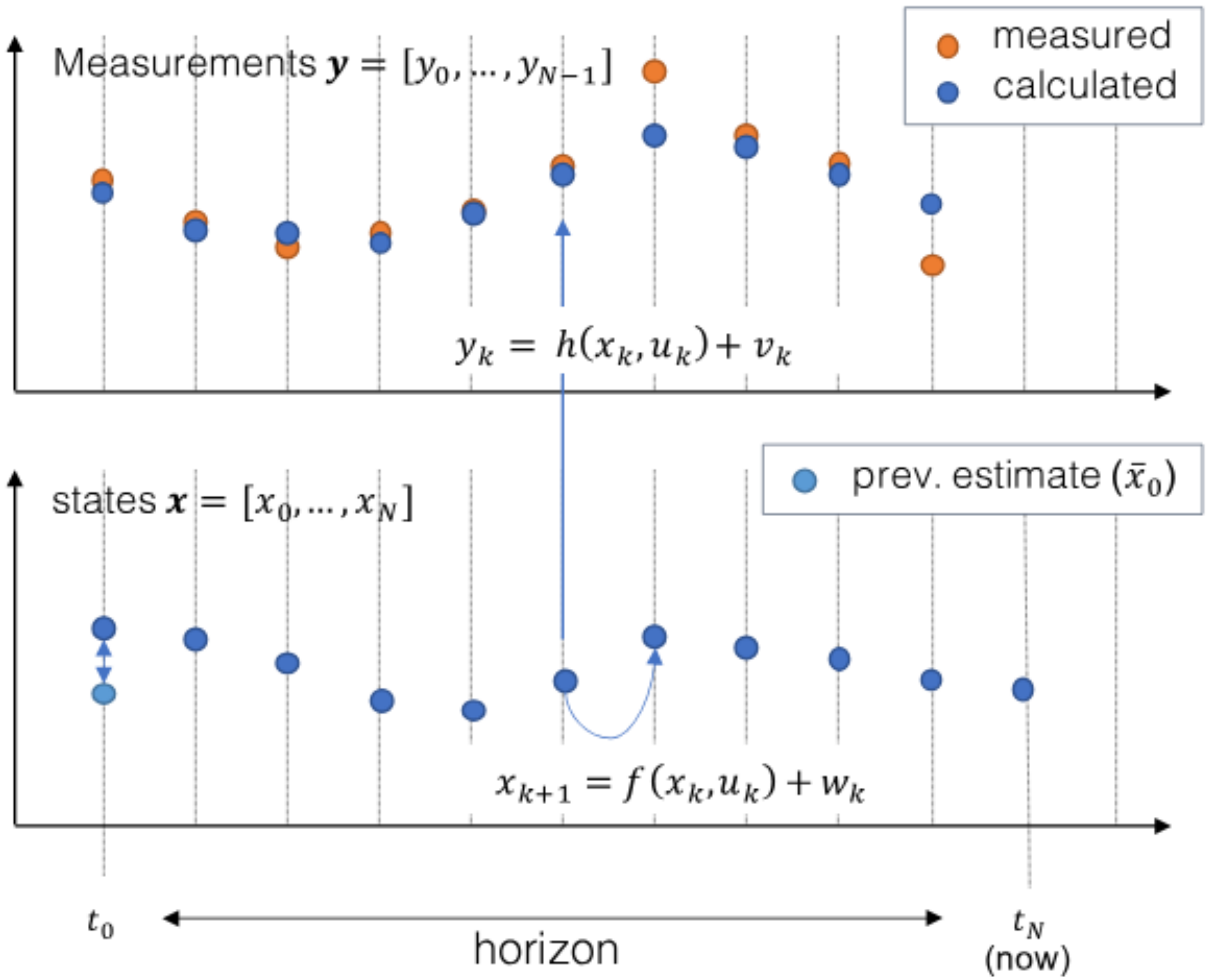
Bring the oscillating masses to a rest

Objective / Cost / Reward:

Use less energy and reach a stable state as soon as possible

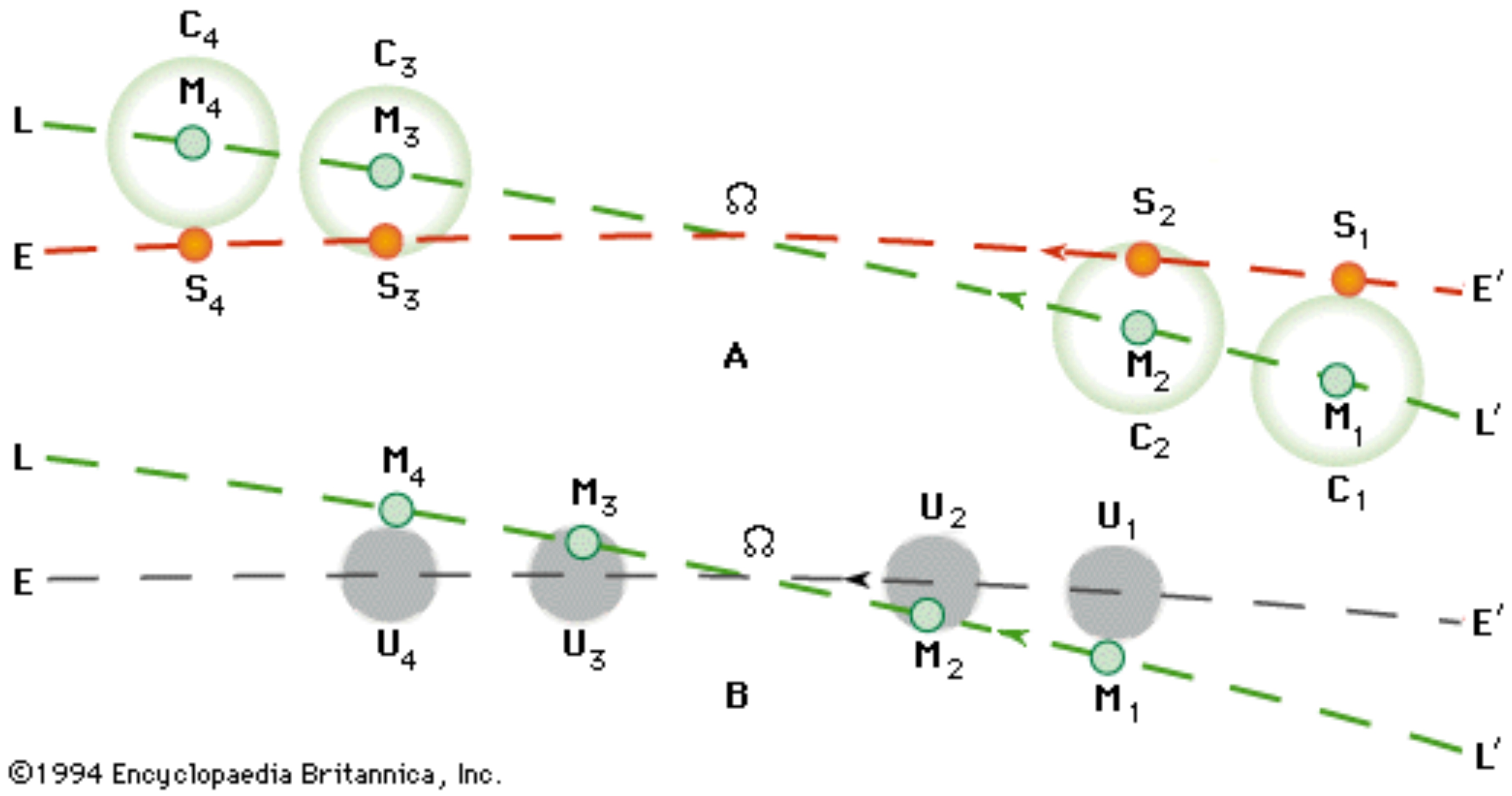
[https://www.do-mpc.com/]

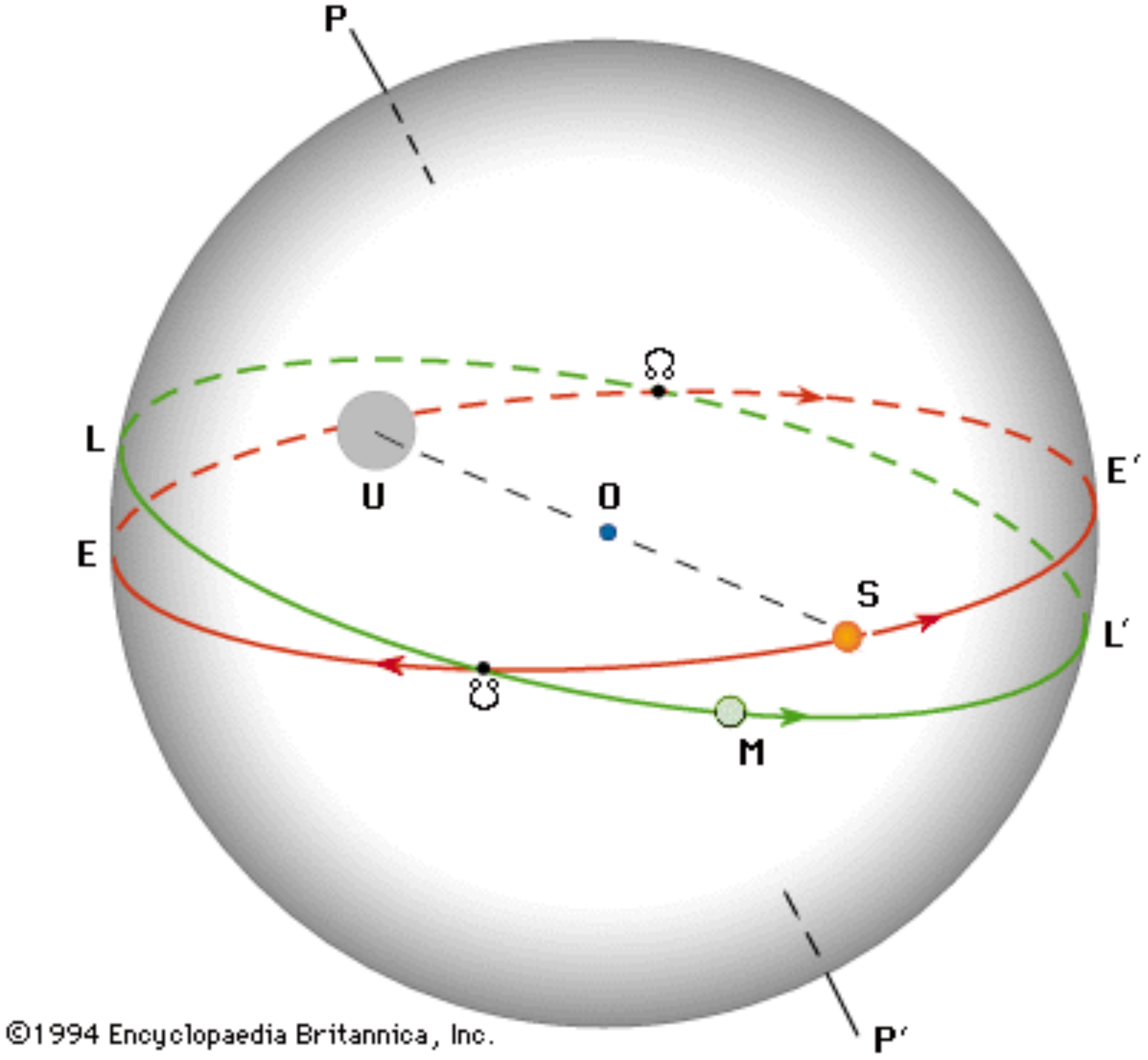
MPC — predictive horizon



[https://www.do-mpc.com/]

Bonus: Eclipse!





Physicists extract useful state variables

Use physical equations to predict

Then calculate when we can reach a desired state

[https://www.britannica.com/science/eclipse/Prediction-and-calculation-of-solar-and-lunar-eclipses]

Learning-based: TD-MPC

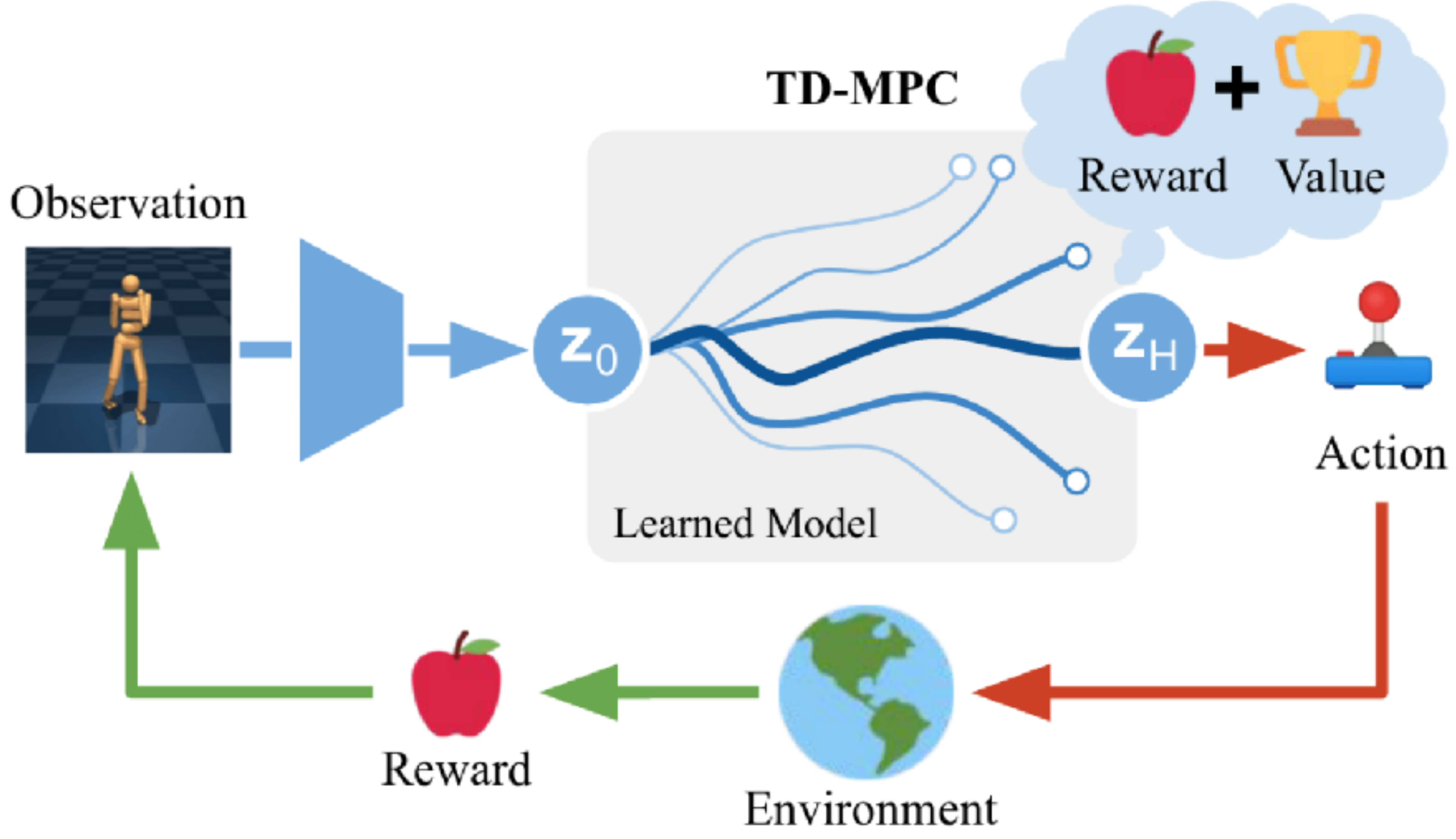
1, Representation: learned state encoder

2, Transition Model: learned MLP

3, Planning algorithm: MPPI (MPC)

It is a continuous version of MuZero by using MPPI instead of MCTS.

Hansen et al. Temporal Difference Learning for Model Predictive Control. ICML 2022.



Learning-based: Diffusion Planner

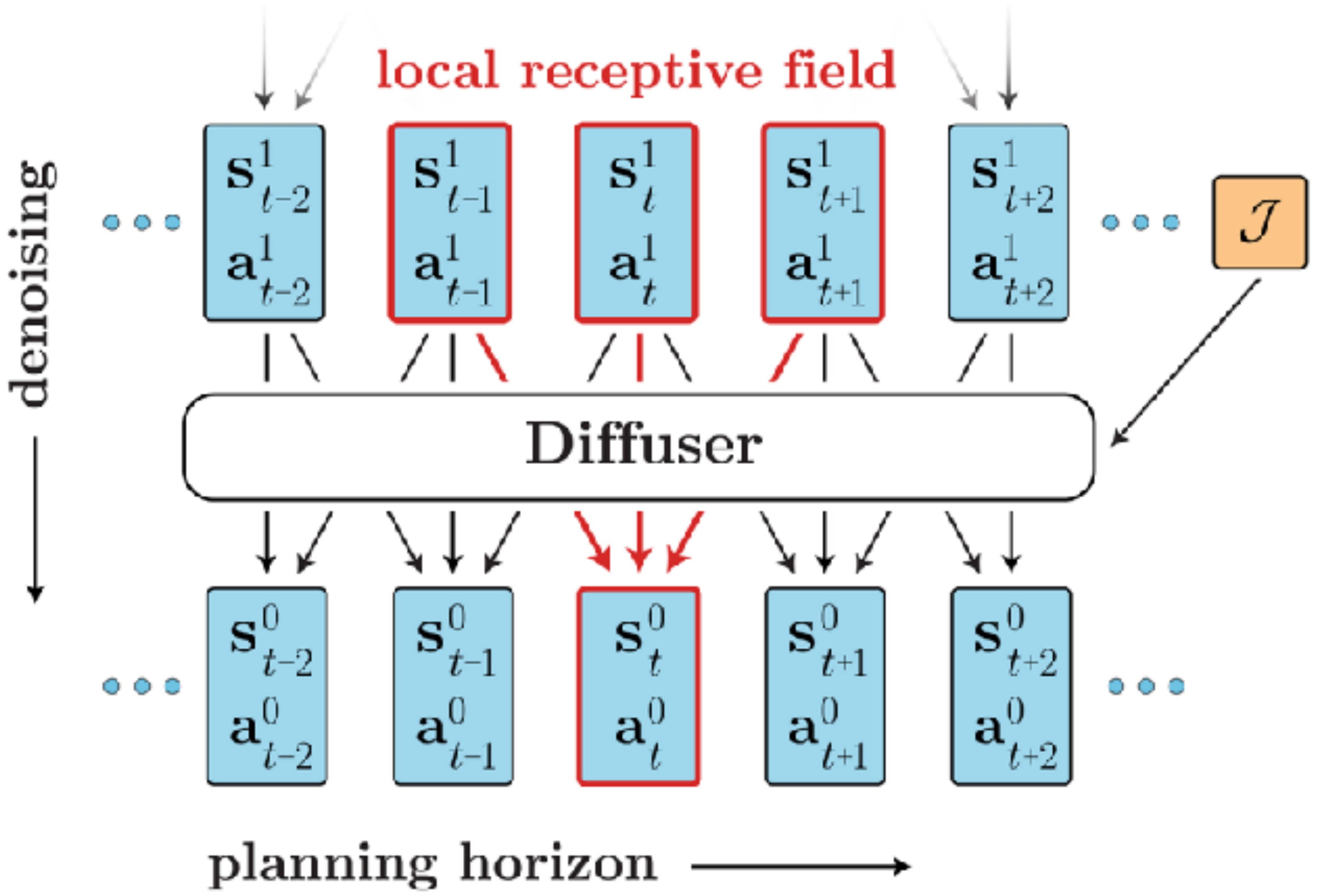
1, Representation: learned state encoder

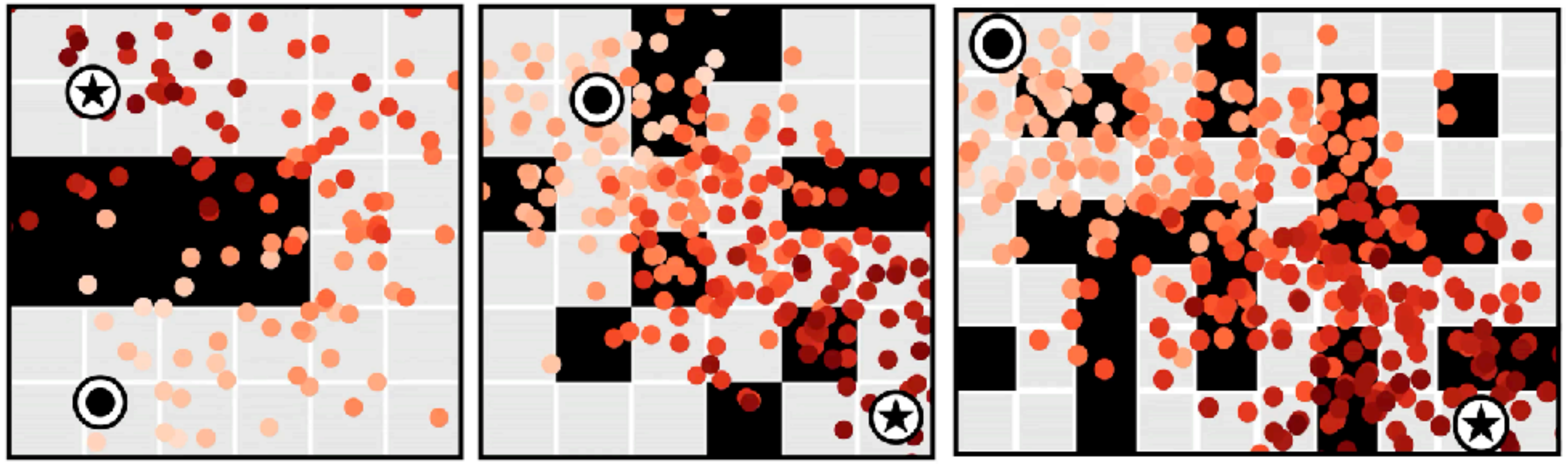
2, Transition Model: learned diffusion model

3, Planning algorithm: learned diffusion model

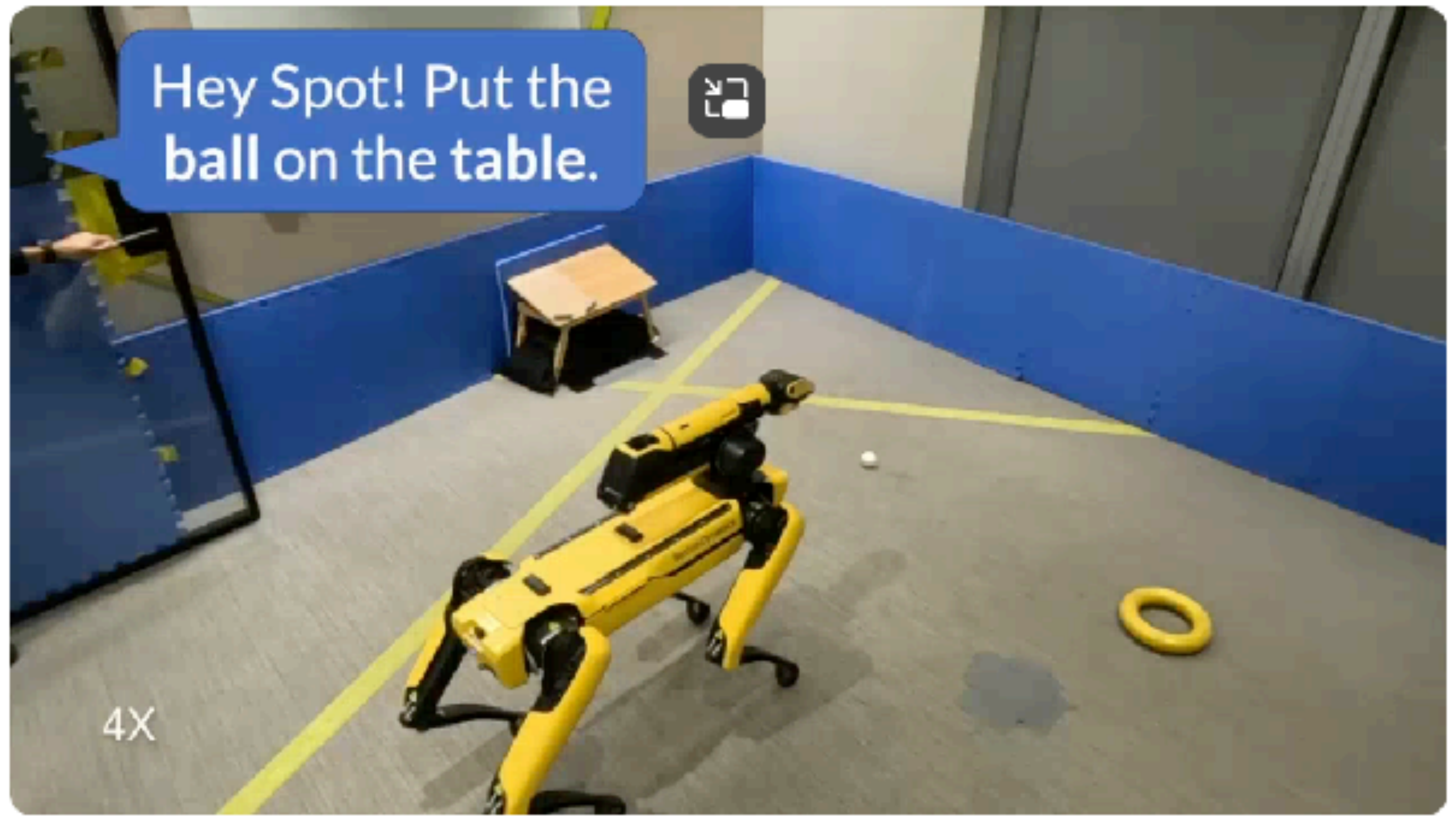
This is a fully end-to-end differentiable architecture

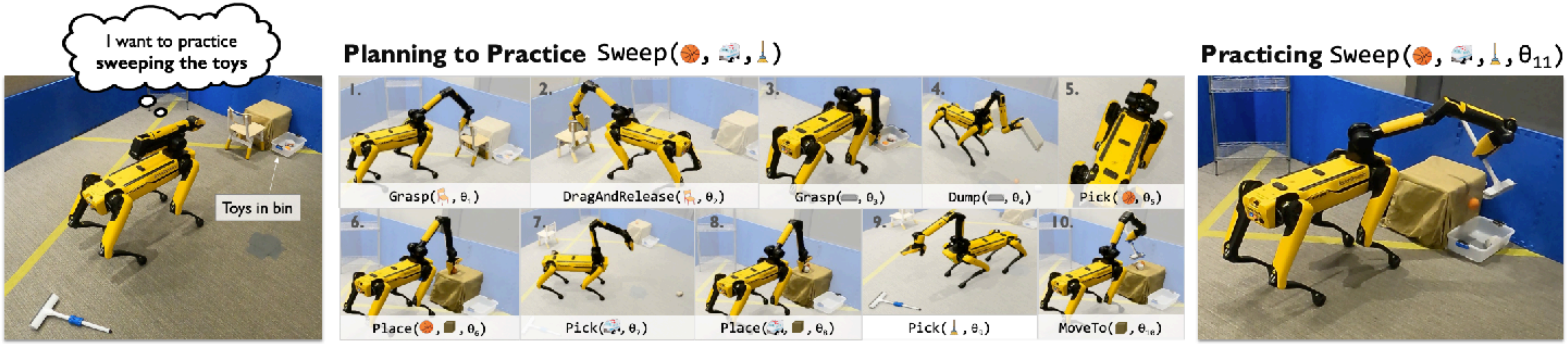
Janner* et al. Planning with Diffusion for Flexible Behavior Synthesis. ICML 2022.



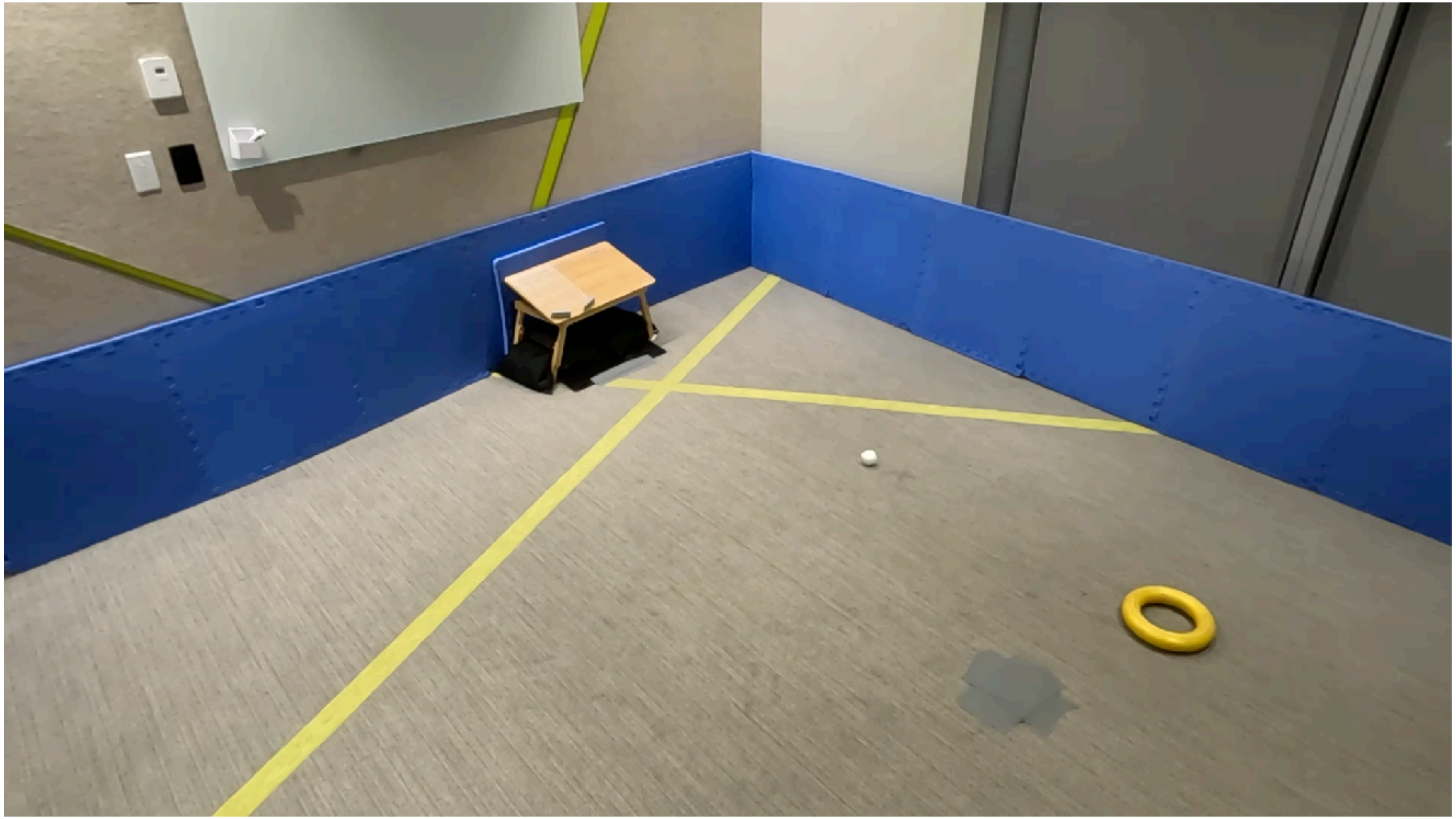


3. Planning in Hybrid Space

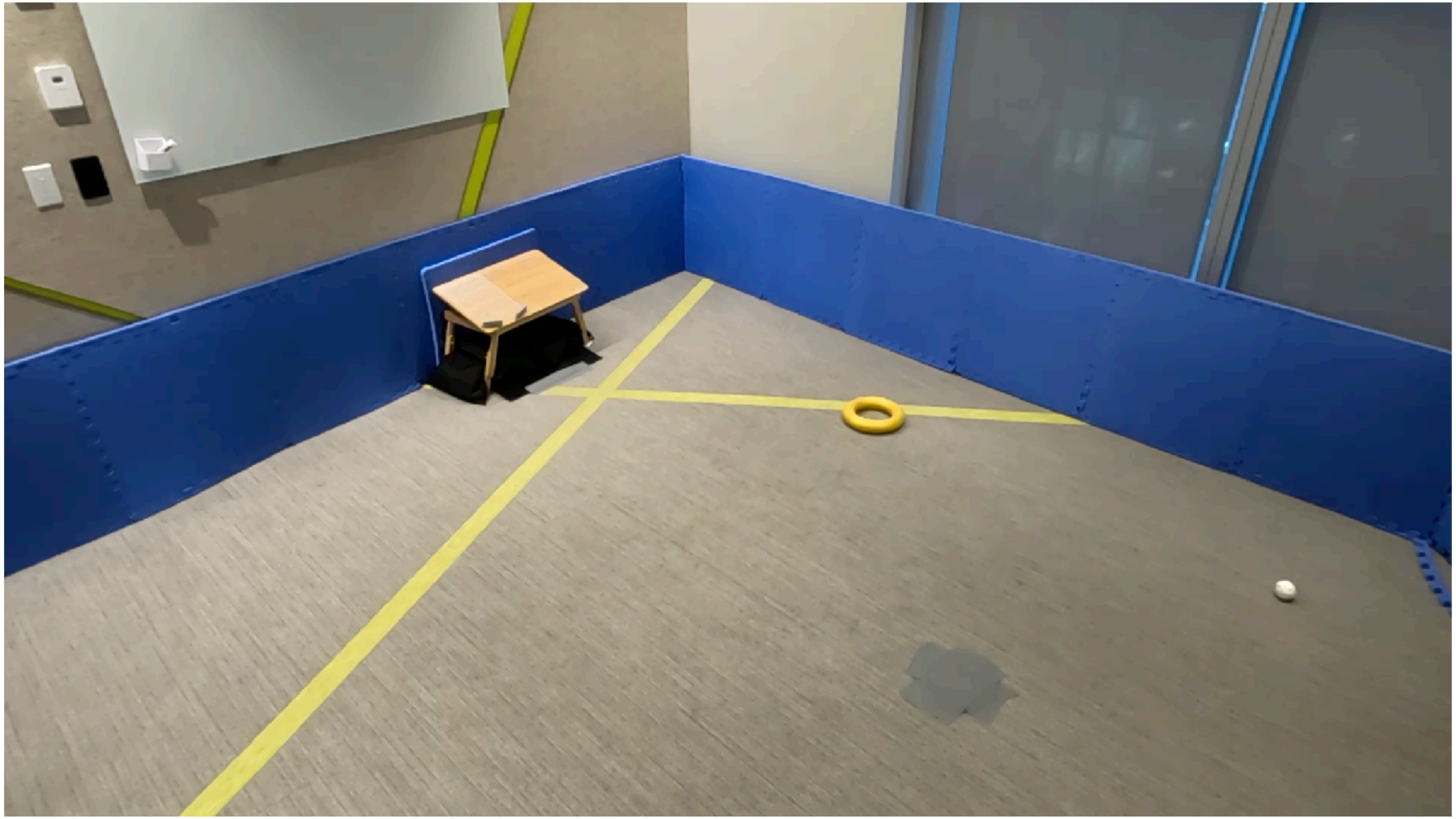




Spot: play ball on table, failure



Spot: place ball on table, success



Integrated Task and Motion Planning

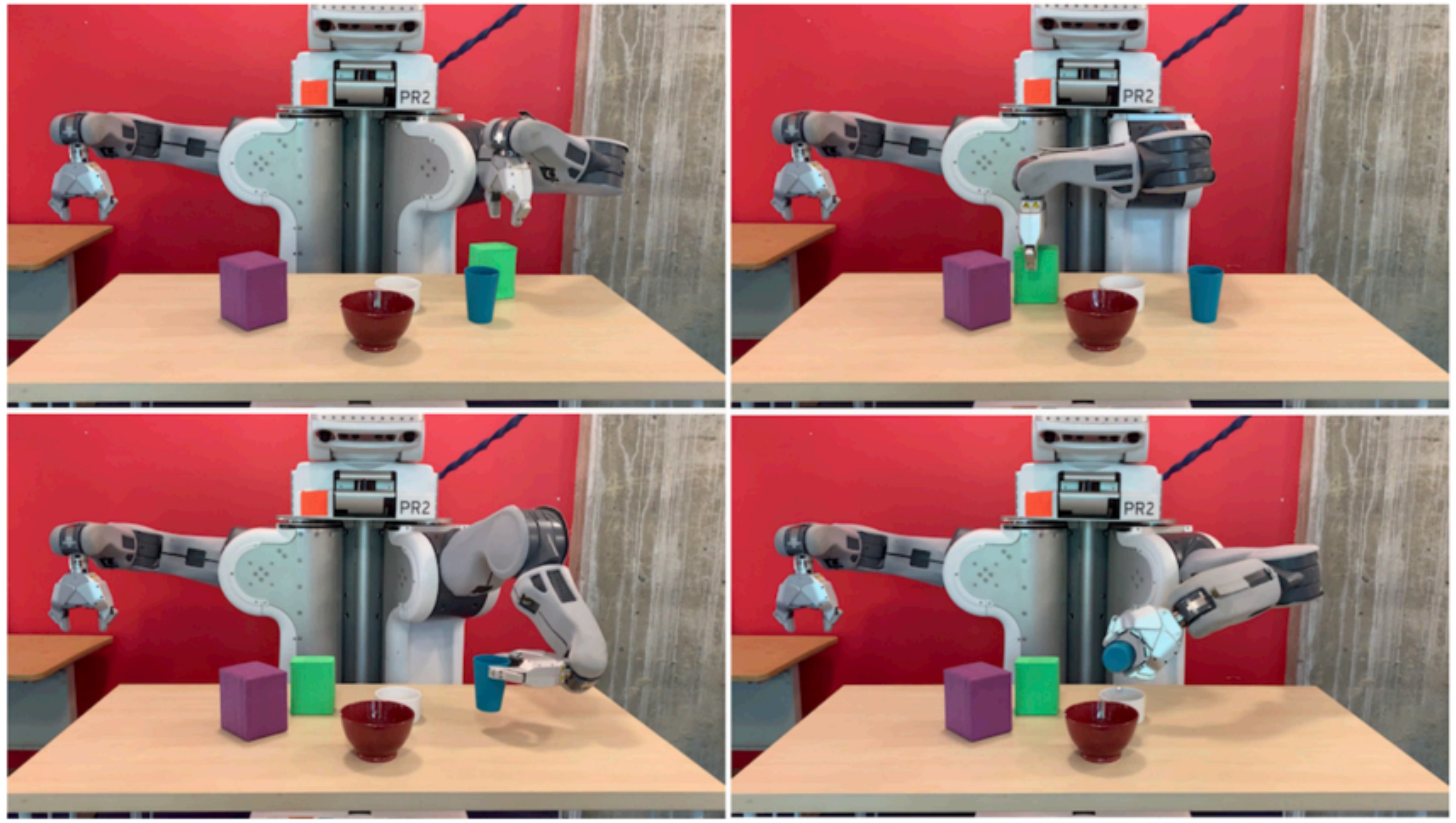


Figure 1: The specified goal is for the contents of the blue cup to end up in the white bowl. Because the green block obstructs reachable grasps for the blue cup, a TAMP algorithm automatically plans to relocate the green block before picking up the blue cup and pouring its contents into the white bowl. From *left-to-right* and *top-to-bottom*: the robot picking up the green block, the robot placing the green block, the robot picking up the blue cup, and the robot pouring the blue cup's contents into the white bowl (8).

3. TASK AND MOTION PLANNING

To find solutions to TAMP problems, we need to integrate aspects of motion planing, multi-modal motion planning, and task planning. In this section, we introduce a framework for describing TAMP problems and algorithms that allows us to describe most of the broad range of existing methods within a unified framework, and which we hope elucidates modeling and algorithmic trade-offs among them. We begin by providing a formalism for describing TAMP problems, then characterize solution methods in terms of their strategies for sequencing actions, for selecting their continuous parameters, and for integrating these methods.

3.1. TAMP problem description

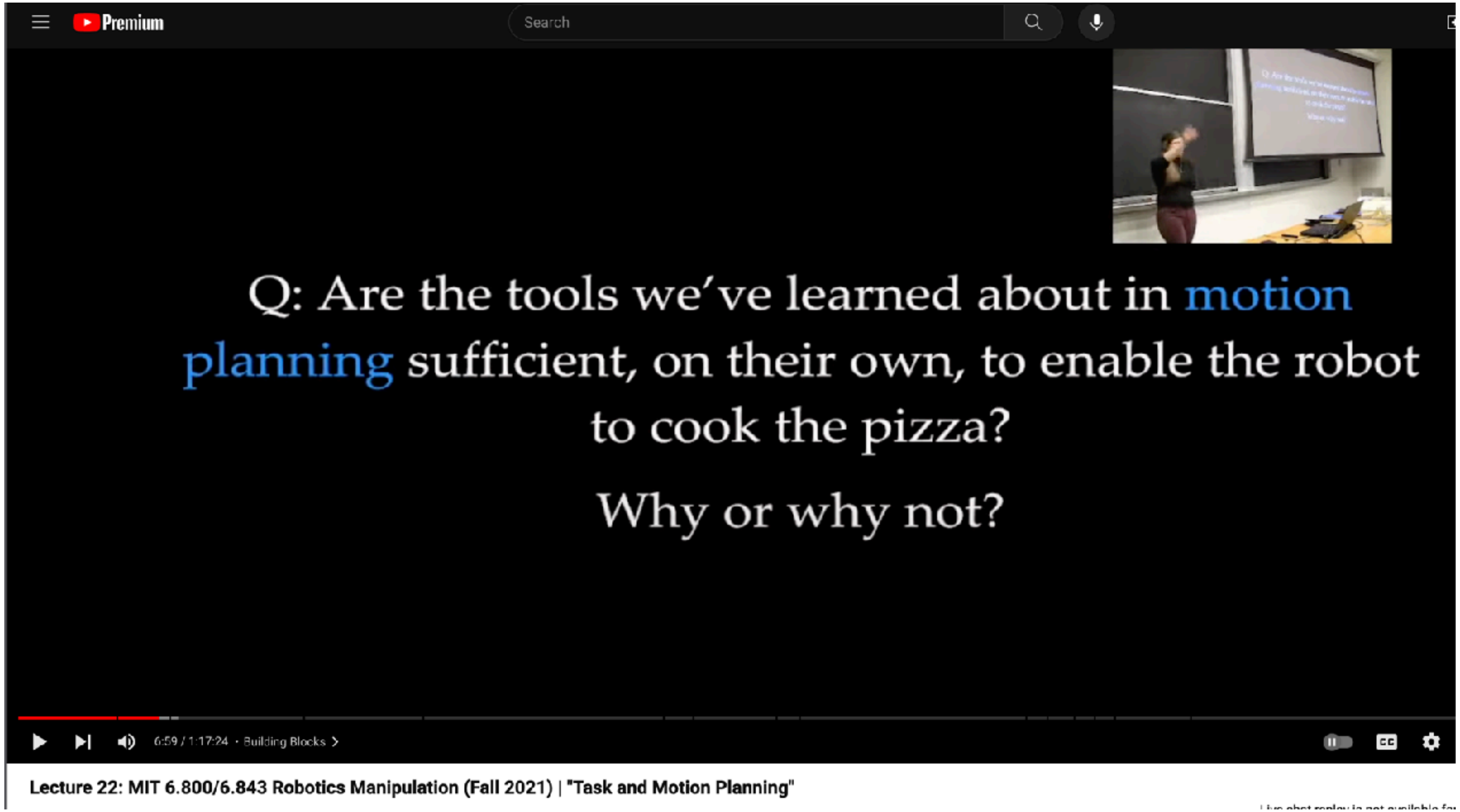
Informally, TAMP problems use compact representational strategies from task planning to describe and extend a class of MMMP problems. TAMP is an extension of MMMP in that there may be additional state variables that are not geometric or kinematic, such as whether the lights are on or the pizza is cooked. We begin by articulating a generic MMMP, using an extension of a task-planning formulation, in Figure 6. There are two extensions of the task-planning formalism visible here. First, there are *continuous* action parameters. Second, in addition to preconditions and effects we have a new type of clause, called **con** for *constraint*. It is a set of constraints that all must hold true among the continuous parameters of the action in order for it to be a legal specification of a transition of the system.

moveWithin[i](θ, w, τ, w')
  con: $\tau(0) = w,\ \tau(1) = w',\ (\forall t \in [0,1]\ F_{\Sigma_i(\theta)}(\tau(t)))$
  pre: mode $= \Sigma_i(\theta)$, **conf** $= w$
  eff: **conf** $\leftarrow w'$
switchModes[i,j](w, θ, θ')
  con: $F_{\Sigma_i(\theta_1)}(w),\ F_{\Sigma_j(\theta_2)}(w)$
  pre: mode $= \Sigma_i(\theta_1)$, conf $= w$
  eff: mode $\leftarrow \Sigma_j(\theta_2)$

Figure 6: A formalization of MMMP in the style of task planning. There is a **moveWithin** action for each mode family Σ_i and a **switchModes** action for each mode family pair Σ_i, Σ_j.

Garrett et al. Integrated Task and Motion Planning. Annual Review of Control, Robotics, and Autonomous Systems. 2021

Why "integrated"?



Lecture: Task and Motion Planning. Rachel Holladay, 2021.

Representative TAMP algorithms

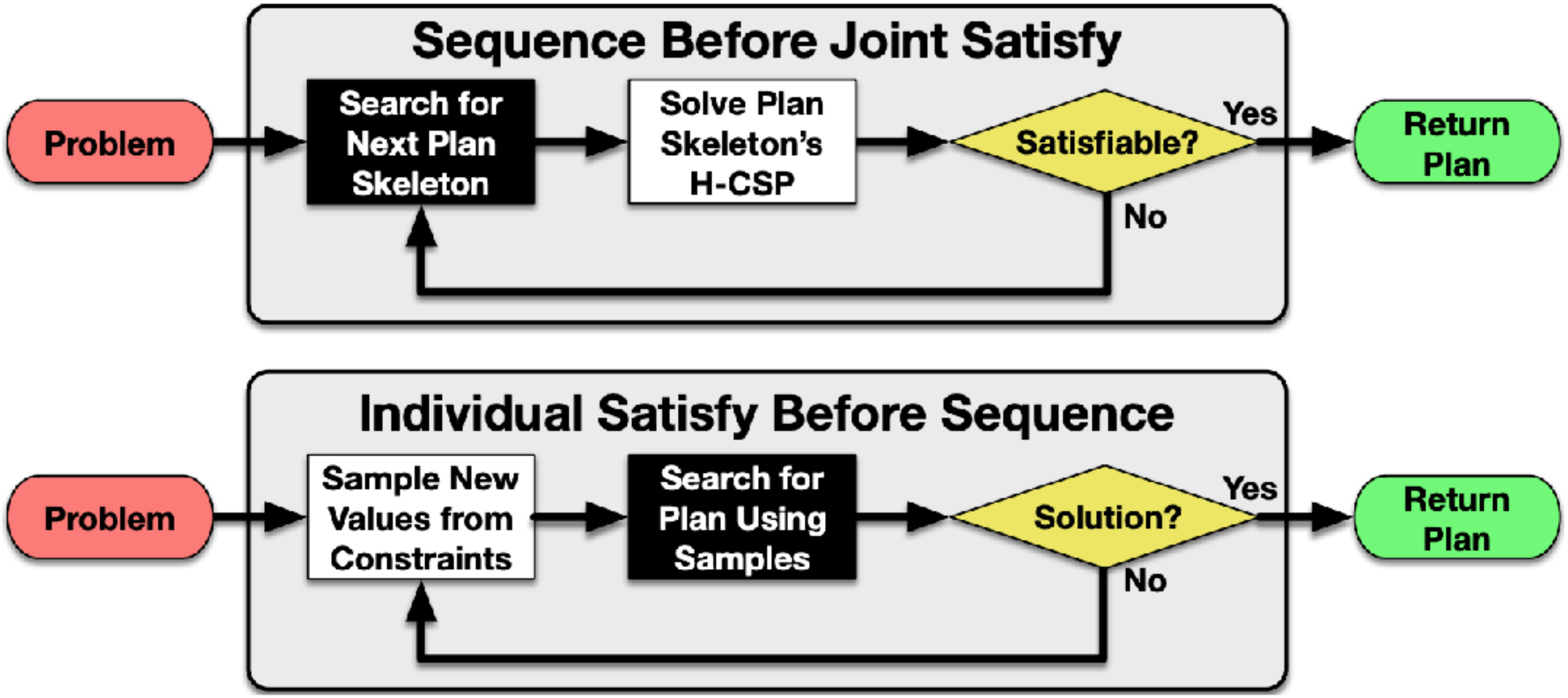


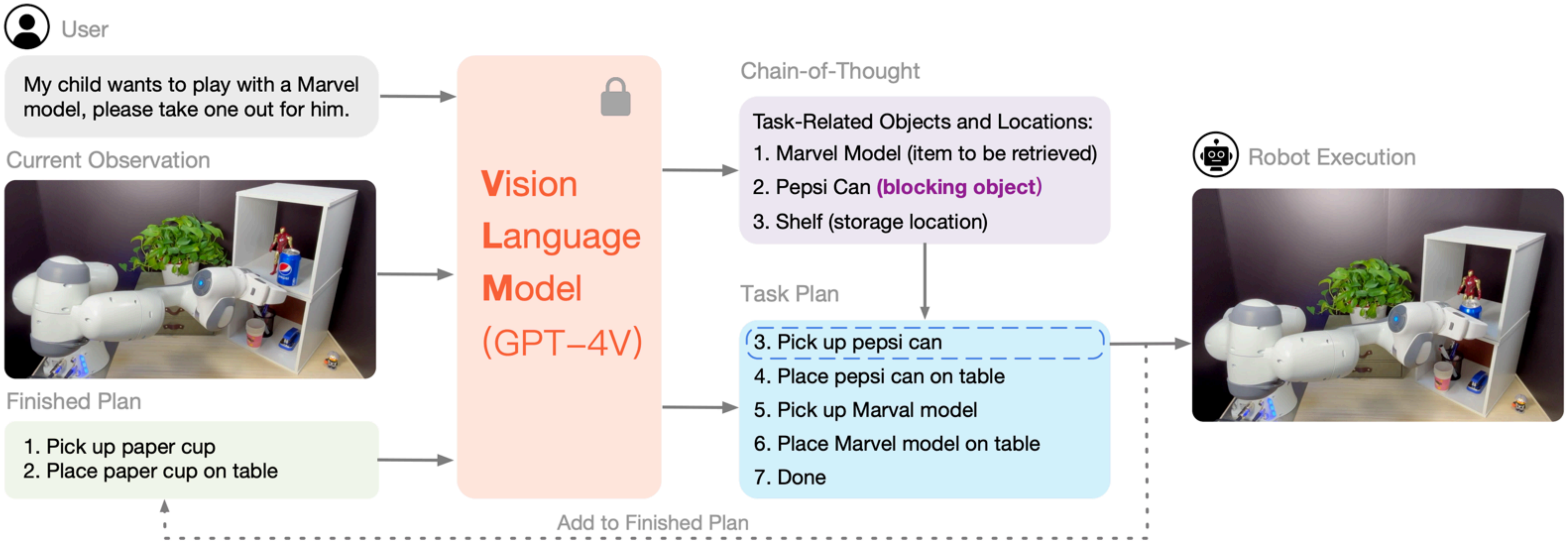
Figure 11: Flowcharts for two representative TAMP algorithms. *Top*: an algorithm that iteratively searches in the space of unbound plans and jointly satisfies the set of constraints (Section 3.3.1). *Bottom*: an algorithm that iteratively performs individual sampling before searching in the space of fully-bound plans (Section 3.3.2).

A table of TAMP approaches (by 2021)

	Pre-discretized	Sampling	Optimization
Satisfaction First	Ferrer-Mestres* (84, 85)	Siméon† (22) Hauser† (13, 29, 14) Garrett* (86, 21) Krontiris† (87, 88) Akbari* (89) Vega-Brown† (90)	
Interleaved	Dornhege* (62, 63, 91) Gaschler* (92, 93, 94) Colledanchise* (95)	Gravot* (96, 97) Stilman† (23, 98, 99) Plaku† (100) Kaelbling* (101, 102) Barry† (103, 30, 104) Garrett* (70, 71) Thomason* (105) Kim* (106, 107) Kingston† (108)	Fernandez-Gonzalez* (109)
Sequence First	Nilsson* (2) Erdem* (74, 75) Lagriffoul* (65, 66, 67) Pandey* (110, 111) Lozano-Pérez* (112) Dantam* (77, 78, 79) Lo* (113)	Wolfe* (114) Srivastava* (76, 60) Garrett* (55, 73)	Toussaint* (61, 68, 69) Shoukry* (81, 82, 83) Hadfield-Menell* (115)

Table 1: A table that categorizes MMMP and TAMP approaches, based on how they solve HC-SPS and how they integrate with constraint satisfaction with action sequencing. Approaches for MMMP are designated with †, and approaches for TAMP are designated with *. Each table cell is listed chronologically.

Using VLMs to Plan



Hu et al. Look Before You Leap: Unveiling the Power of GPT-4V in Robotic Vision-Language Planning. arXiv 2023.

**This paper has been using teleop for real-robot demo*

Outline

Goals and Motivation

Basics of Planning

The Role of Learning in Planning

Planning Algorithms & Integration with Learning

Case Study: Mobile Manipulation

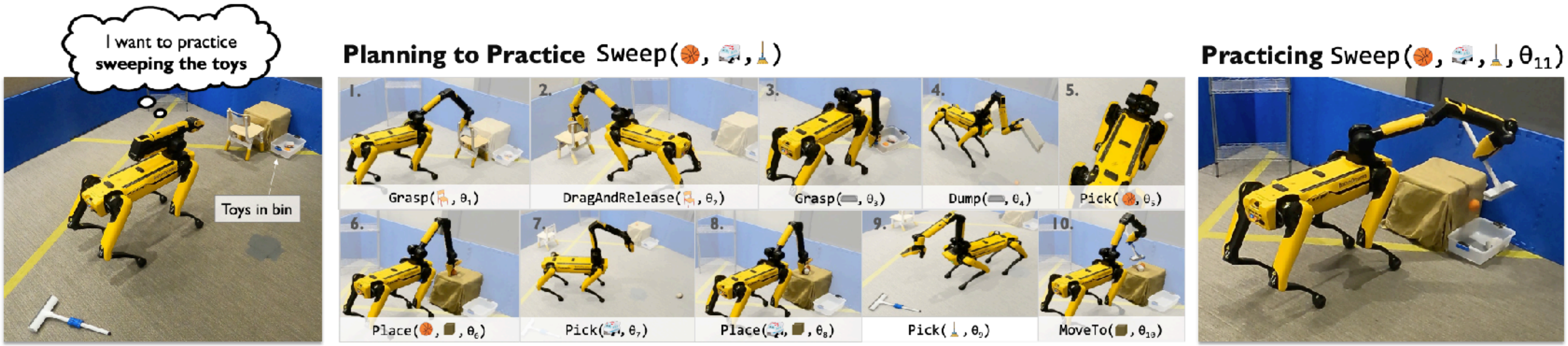
Takeaways

Case Study:
Mobile Manipulation with Spot

Spot: From raw camera to motor actions

[Kumar*, Silver*, McClinton, Zhao, Proulx, Lozano-Perez, Kaelbling, Barry. Under Review 2024]

Challenges



Still far from general-purpose robots with long-horizon planning for mobile manipulation

- We don’t have a model or even data for training one on real robots!

- Partial observability, action execution noise, accurate skills…

Overview: Bilevel Planning

Idea:

Build a high-level symbolic model

Hand design skills/operators

Use AI planner to solve high-level planning problem

Then ground symbolic actions to physical world

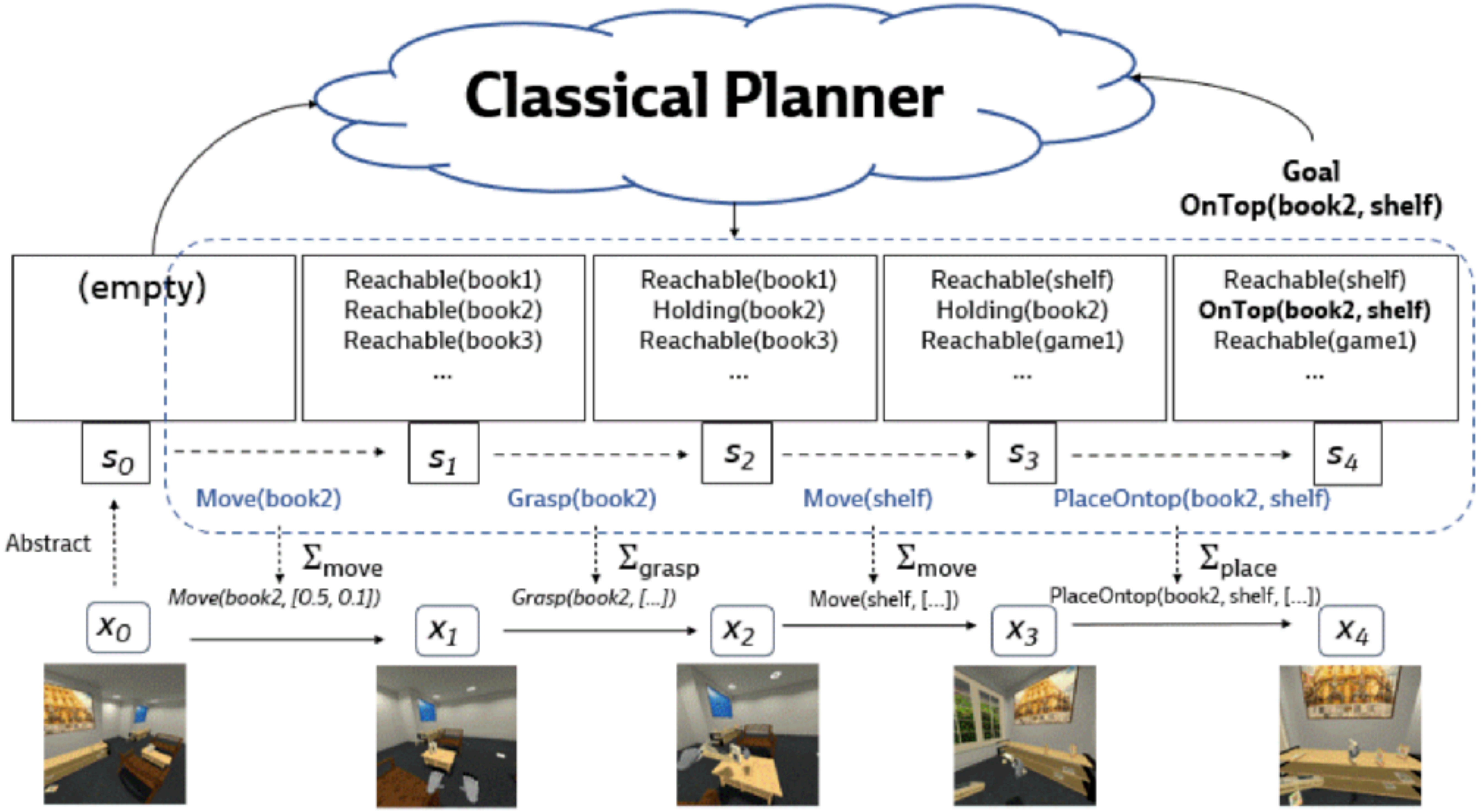


Figure 8: Animated visualization of constructing an abstract plan, and then 'refining' this plan using samplers (denoted by ∑) to derive the continuous parameters for skill associated with an operator. These skills now have all their parameters specified, so can be executed in the environment in sequence.

Specifying Skill Operators

Arguments

List of typed variables

Preconditions

What must be true in order to use this operator?

Add/Delete Effects

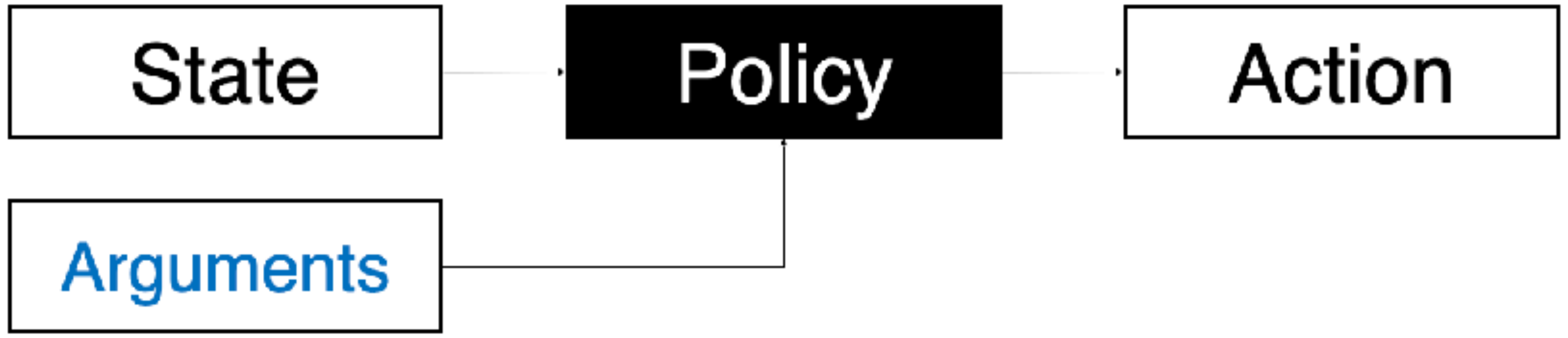
How is the abstract state changed by this operator?

```
Operator-PickFromTable:

Arguments: [?b - block, ?r - robot]

Preconditions: {GripperOpen(?r),
                OnTable(?b)}

Add effects: {Holding(?b)}
Delete effects: {GripperOpen(?r),
                 OnTable(?b)}
```

[Credit: Tom Silver]

How should I get there?

Skill **Policies**



Same as operator

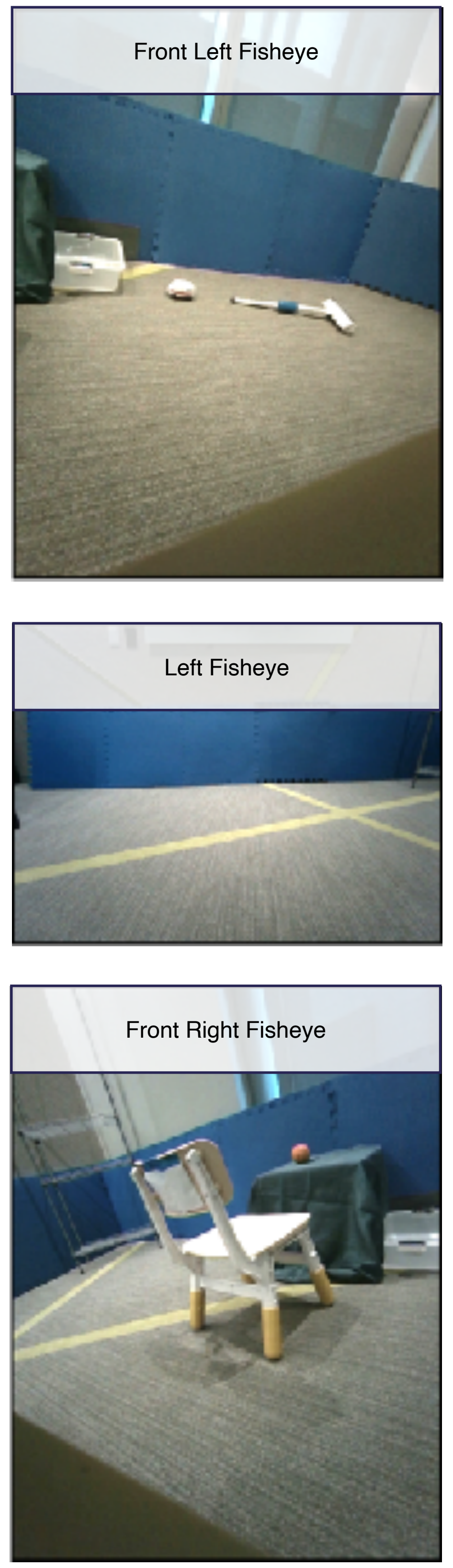
```
def policyPickFromTable(state, ?b, ?
r):
    dx = (state[?b].x - state[?r].x)
    dy = (state[?b].y - state[?r].y)
    dz = (state[?b].z - state[?r].z)
    return [dx, dy, dz]
```

Simplified example

The policy should *achieve* the operator effects when the operator preconditions hold

Perception / State Representation

RGBD
(On-Robot Cameras)



...

Prompts
(Manually-defined)

scrubbing brush/
hammer/
mop/
giant white toothbrush

small white ambulance
toy/
car_(automobile) toy/
egg

white plastic container with
black handles/
white plastic tray with
black handles/
white plastic bowl/
white storage bin with
black handles

small white ball/
ping-pong ball/
snowball/
cotton ball/
white button

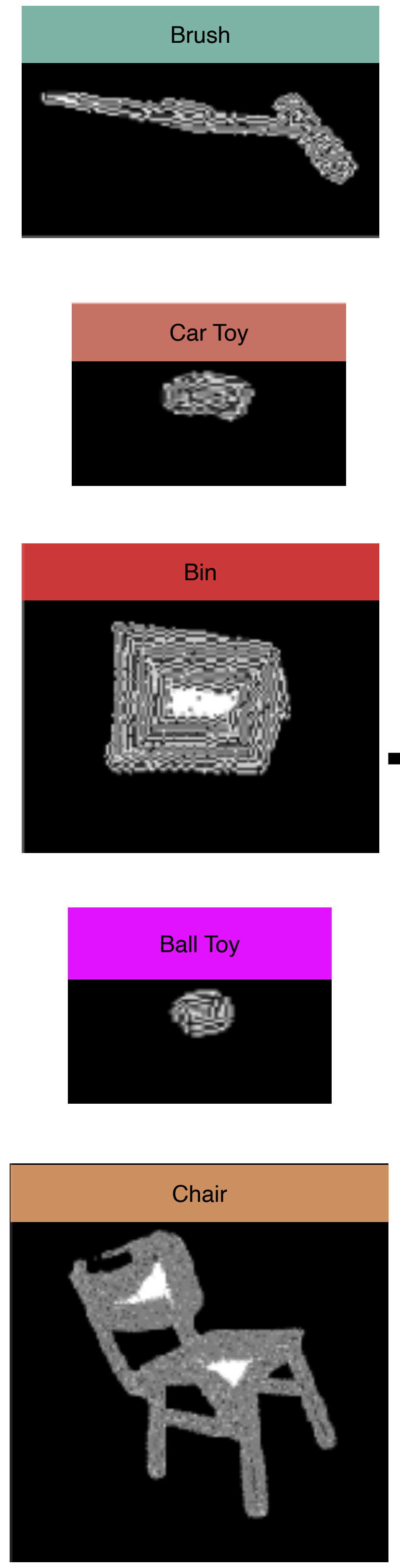
chair

...

Detic
(Zhou et al. 2022)

...

SAM
(Kirillov et al. 2023)



...

Object-Centric State

id	type	x	y	z	...
brush	movable	7.44	5.25	0.32	...

id	type	x	y	z	...
car toy	movable	6.85	5.31	0.31	...

id	type	x	y	z	...
bin	container	6.52	5.40	0.33	...

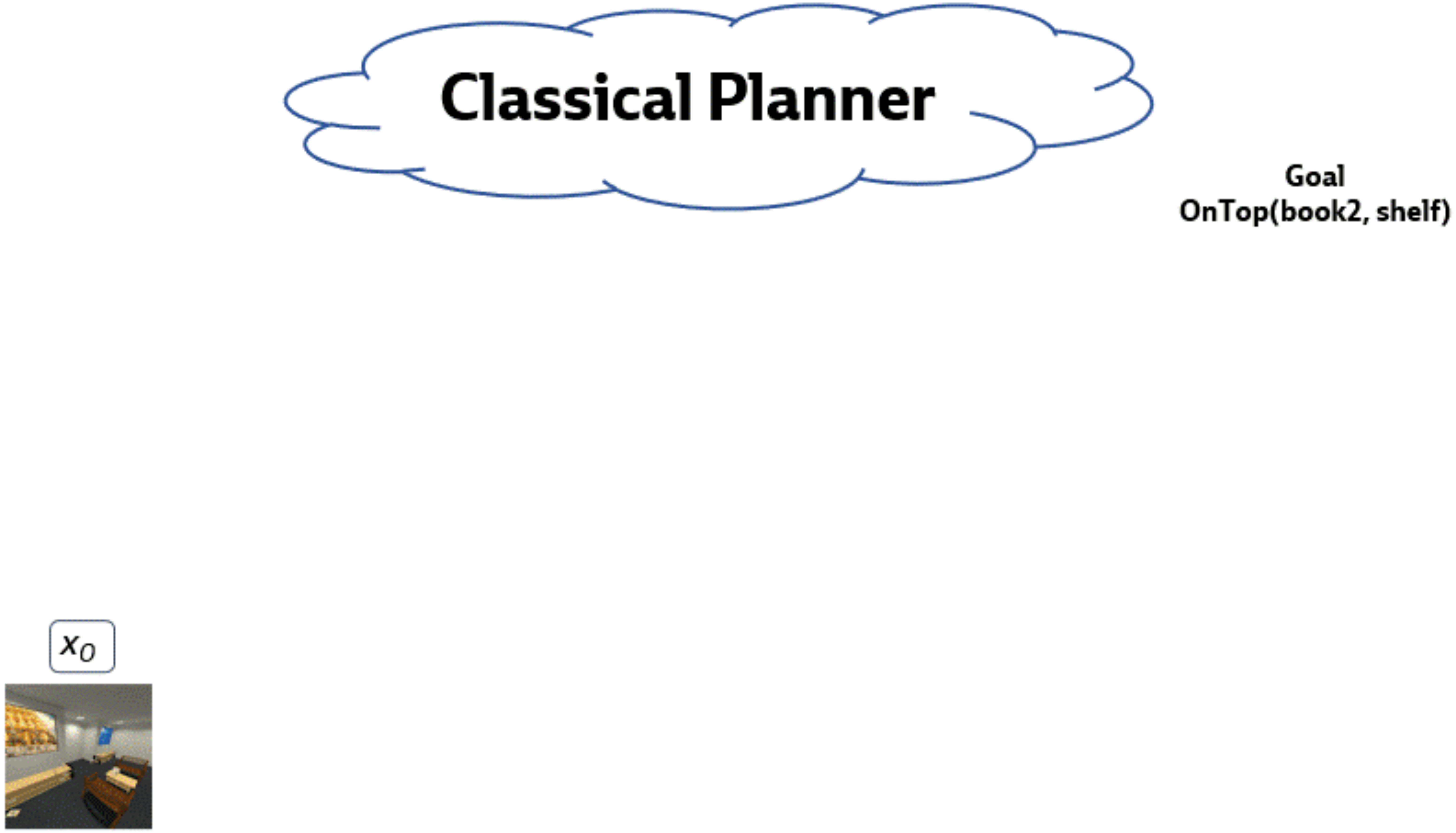
id	type	x	y	z	...
ball toy	movable	5.44	5.61	1.21	...

id	type	x	y	z	...
chair	movable	5.39	4.98	0.55	...

...

[Kumar*, Silver*, McClinton, Zhao, Proulx, Lozano-Perez, Kaelbling, Barry. Under Review 2024]

Bilevel Planning demonstration




8X

Outline

Goals and Motivation

Basics of Planning

The Role of Learning in Planning

Planning Algorithms & Integration with Learning

Case Study: Mobile Manipulation

Takeaways

Takeaways

Long-horizon planning is a very challenging problem, particularly for planning on robots

It involves approximating complex functions: abstracting states and actions, modeling the world, planning on high-dimensional space

Learning needs to be well integrated with planning, so the planning algorithms could scale up to complicated, raw sensor-input, long-horizon tasks

Thank you!

Goals and Motivation

Basics of Planning

The Role of Learning in Planning

Planning Algorithms & Integration with Learning

Case Study: Mobile Manipulation

Takeaways

Sources / References

- Reinforcement Learning: An Introduction. Andrew Barto and Richard S. Sutton 2018.
- Lecture: Integrated learning and planning. David Silver, 2015.
- Lecture: Optimal Control and Planning & Model-based RL. Sergey Levine, 2017.
- A Theory of Abstraction in Reinforcement Learning. David Abel, PhD Thesis 2020.
- Planning Algorithms. Steven M. LaValle, 2006.
- Bilevel Planning for Robots: An Illustrated Introduction. Nishanth Kumar, Willie McClinton, Kathryn Le, Tom Silver. MIT LIS Blog 2023.
- (Presentation slides) Kumar*, Silver*, McClinton, Zhao, Proulx, Lozano-Perez, Kaelbling, Barry. Practice Makes Perfect: Planning to Learn Skill Parameter Policies. Under Review 2024.