Carnegie Mellon

School of Computer Science

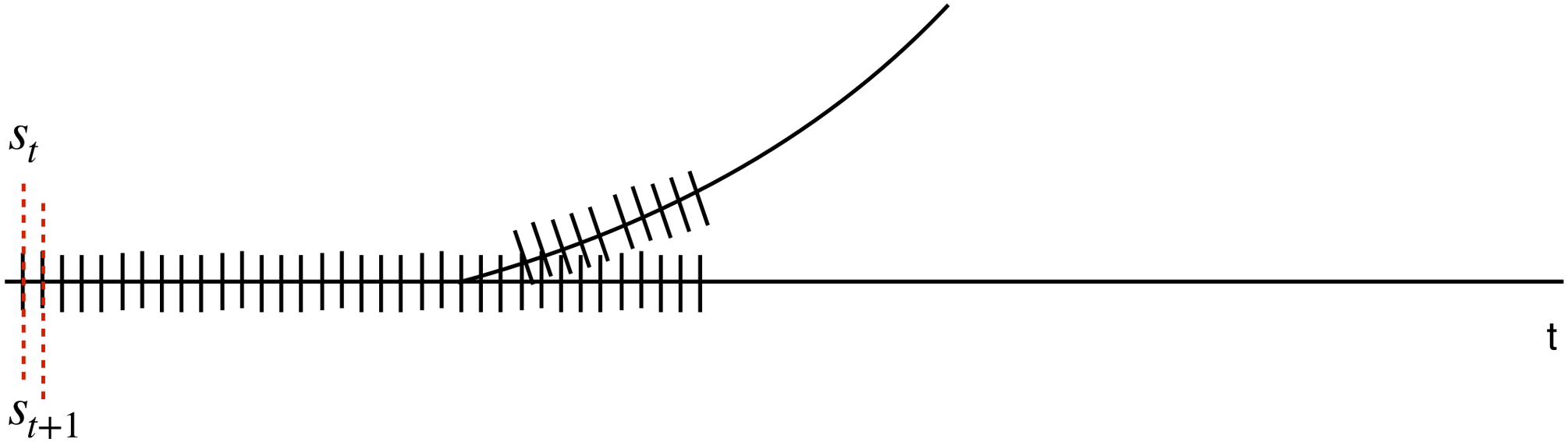
Deep Reinforcement Learning and Control

Learning and Planning: Temporal abstraction in model based control

Spring 2020, CMU 10-403

Katerina Fragkiadaki

Model-based control



Predicting fine-grained state transitions $s_{t+1} \sim f(s_t, a_t)$ is hard:

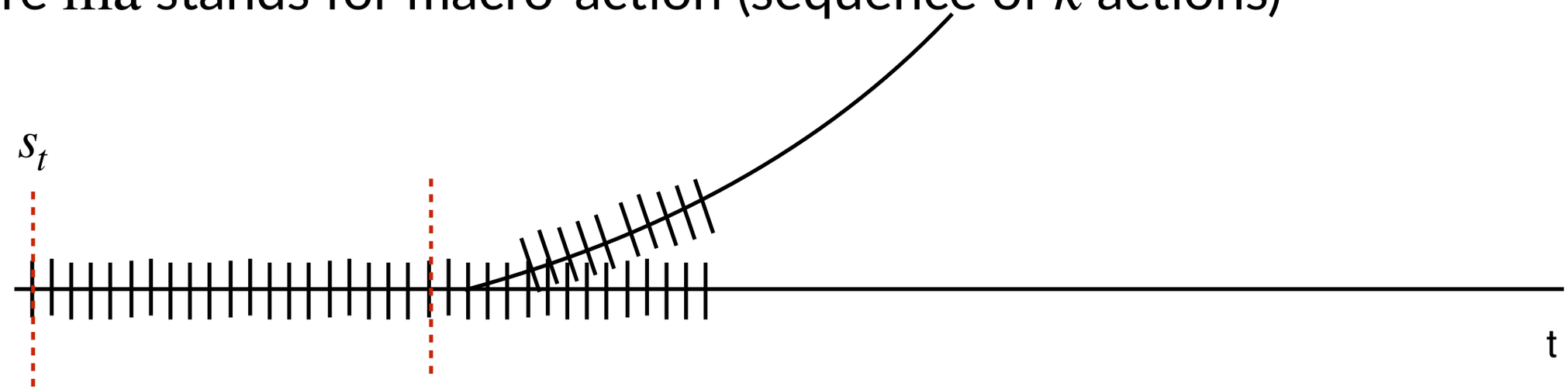
1. there is a lot of uncertainty/ stochasticity during contact
2. errors accumulate over time

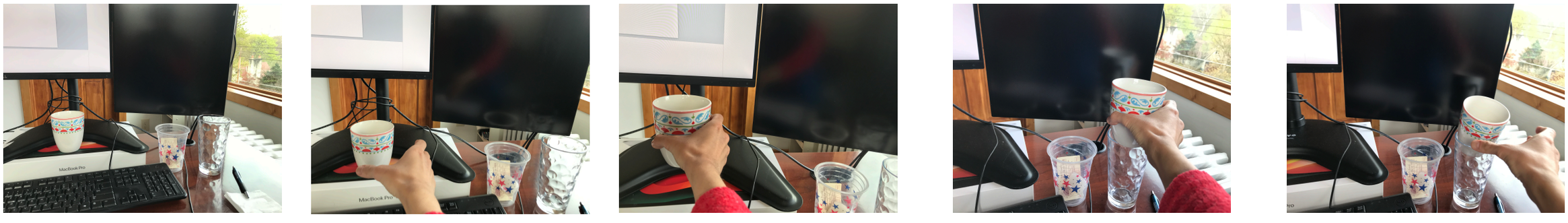
How do we humans imagine consequences of our actions?

Model-based control with temporal abstraction

Predict environment transitions in coarser time scales $s_{t+k} \sim f(s_t, \mathrm{ma}_t)$

where ma stands for macro-action (sequence of k actions)





- Behaviours of shorter than k timesteps: no look-ahead (reactive learning)
- Behaviours longer than k timestpes: look-ahead (planning)

We have no planning happening within each macro-action.

We have planning (model look-ahead) across macro-actions.

In general macro-actions can have various lengths

How can we learn macro-actions?

Macro-actions

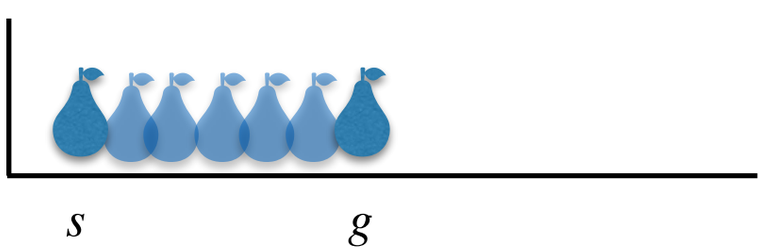
A sequence of actions that takes you from one state to another.

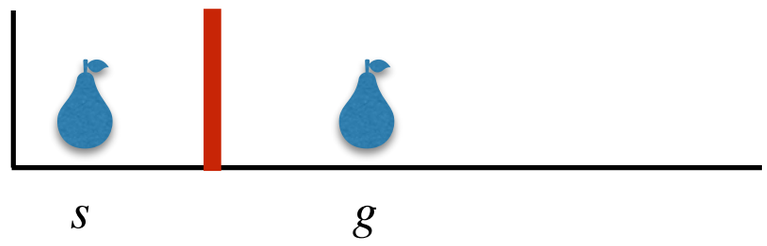
In general I need macro actions that can reach any state that is k steps apart.

- Goal-conditioned policies: $\pi(s, g; \theta)$ (similar to multistep inverse models)

A problem with the goal conditioned policies (the way we have presented thus far) is that:

- They do not expose the trajectory followed
- They do not learn multiple trajectories from start s to goal g





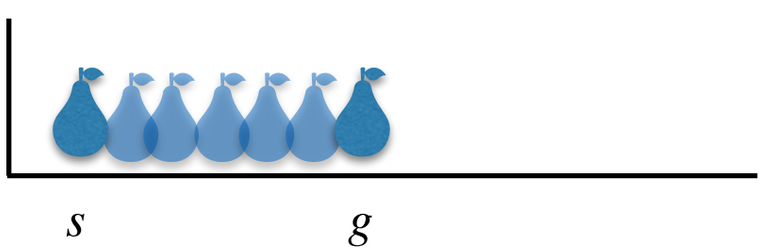
Often i wish to prefer one way of going from s to g over another

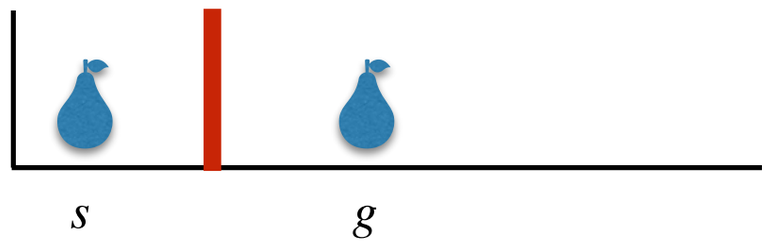
Macro-actions

A sequence of actions that takes you from one state to another.

In general I need macro actions that can reach any state that is k steps apart.

- Goal-conditioned policies: $\pi(s, g; \theta)$ (similar to multistep inverse models)
- Plan conditioned trajectories: $\pi(s, z; \theta)$

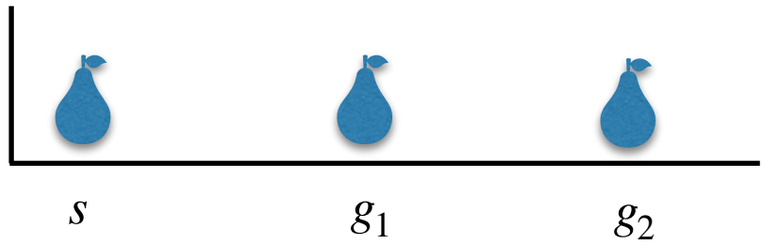


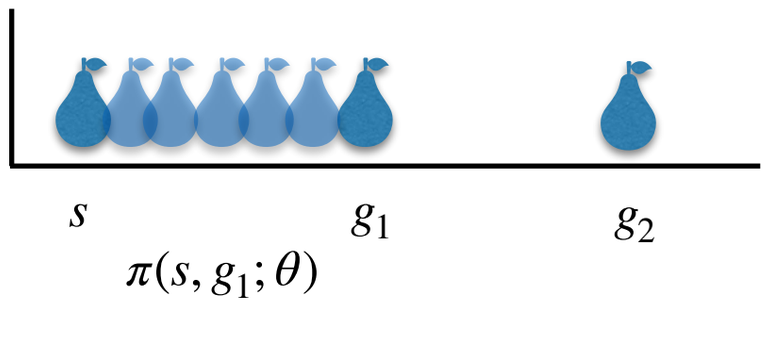


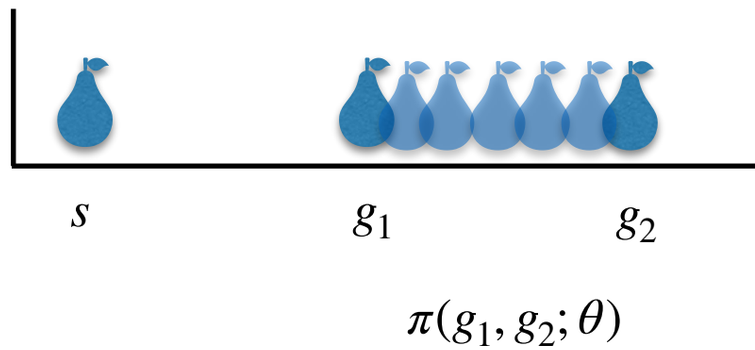
Often we prefer one way of going from s to g over another, and we want control over such choices.

Why planning and learning?

Notice the translation invariance of behaviors





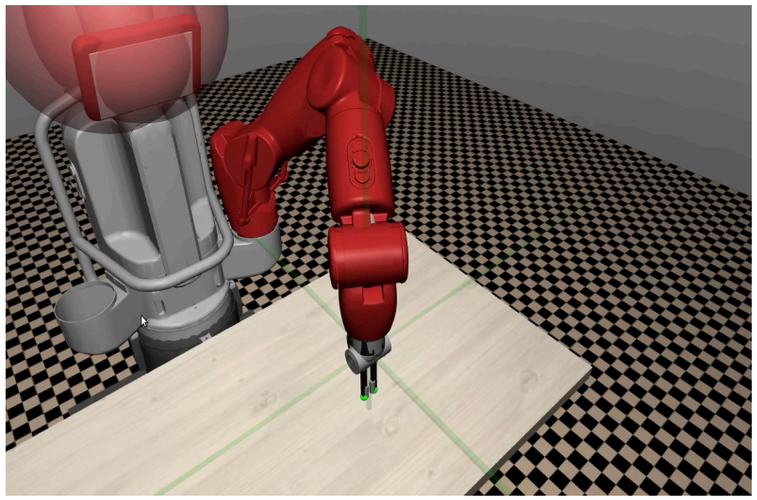


The two skills: $\pi(s, g_1; \theta)$ and $\pi(g_1, g_2; \theta)$ should be very similar, thus, learning $\pi(s, g_1; \theta)$ should permit me by composition to easily (without much exploration) achieve many other states

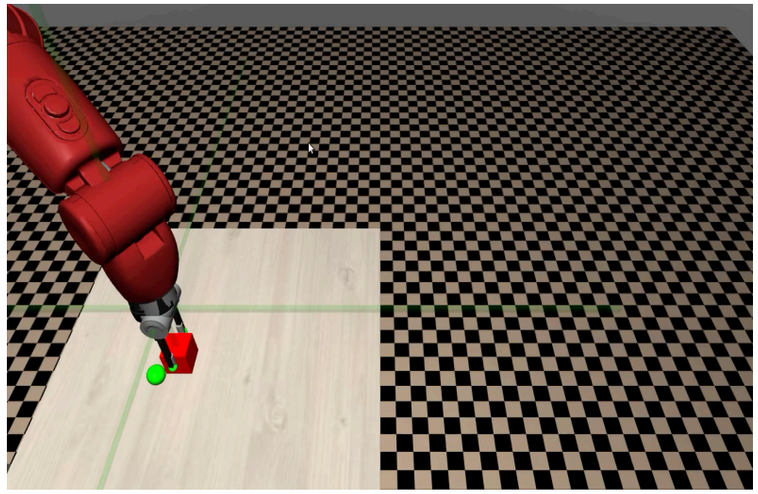
Learning the primitives: goal-conditioned policies

We will first learn a set of skills, i.e., we will manually design the goal set for each and the reward functions (standard model-free RL, e.g., HER+DDPG)

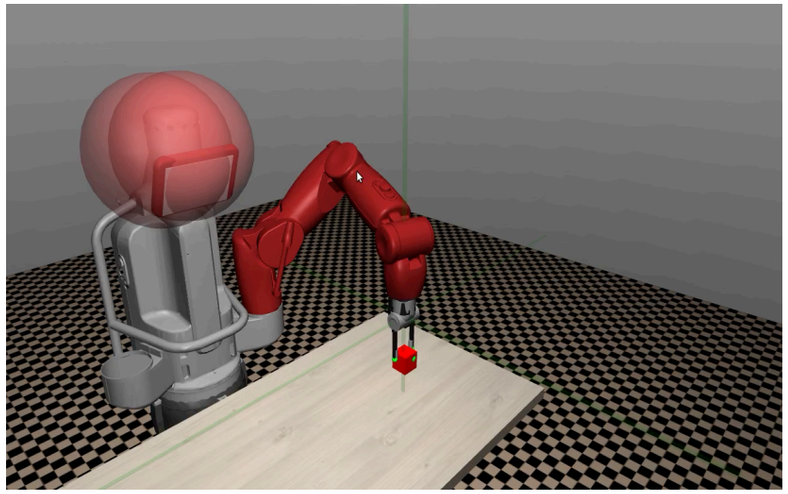
Transit



Grasping



Transfer



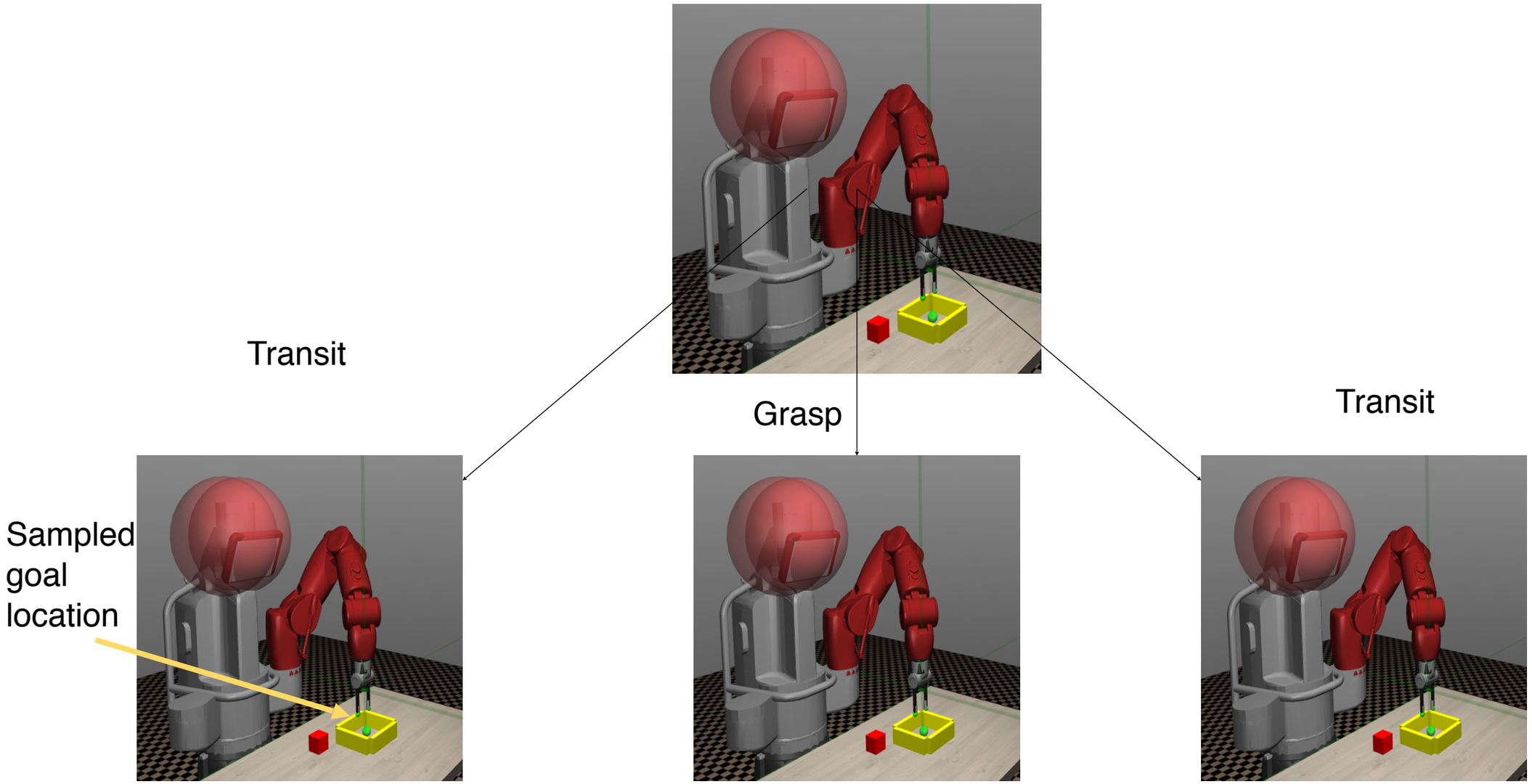
To use the skills for look-ahead we also need to learn:

- a coarse transition model for each $s' = f^i(s, g), i = 1...3$, in case the skill does not exactly take you to goal g
- a success model $W^i(s, g) \rightarrow [0,1], i = 1...3$

Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

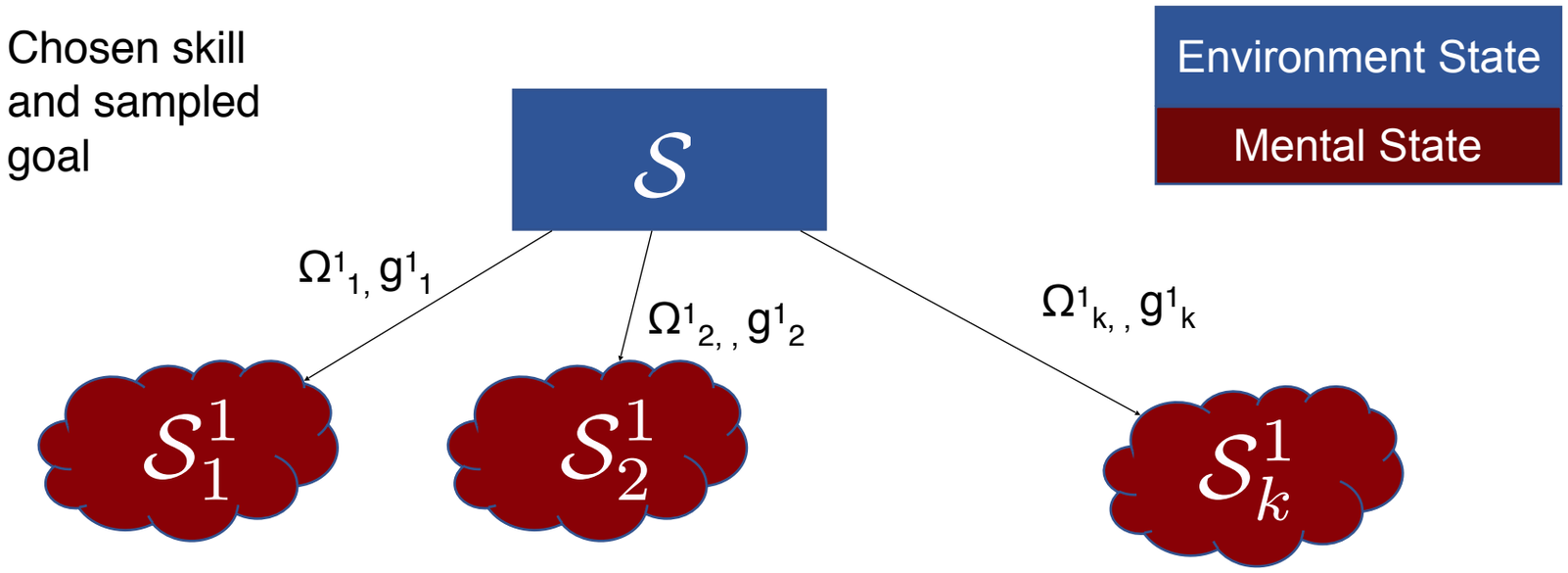
Planning using goal-conditioned policies

To look-ahead we need to sample the skill and its goal parameter:



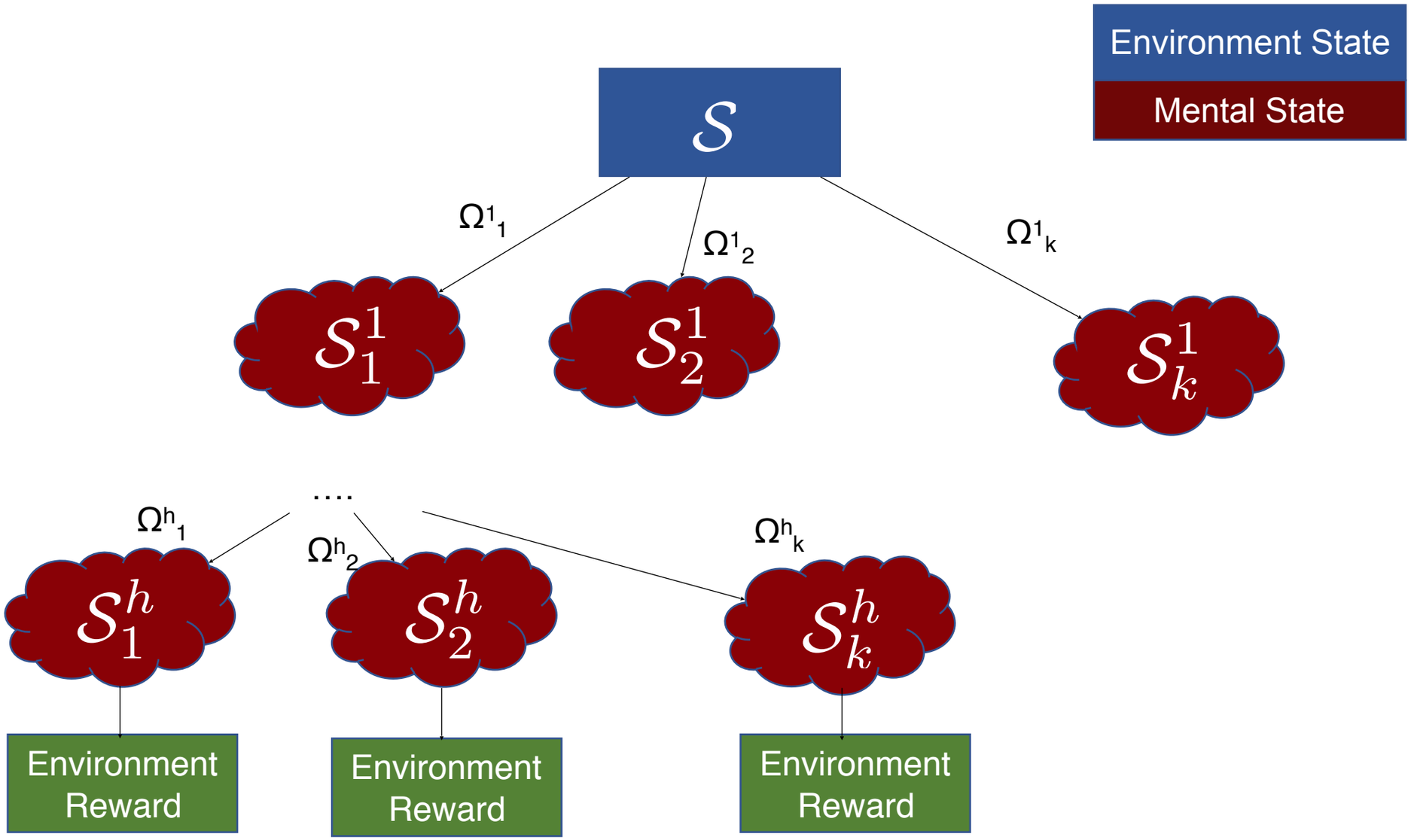
Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Planning using goal-conditioned policies



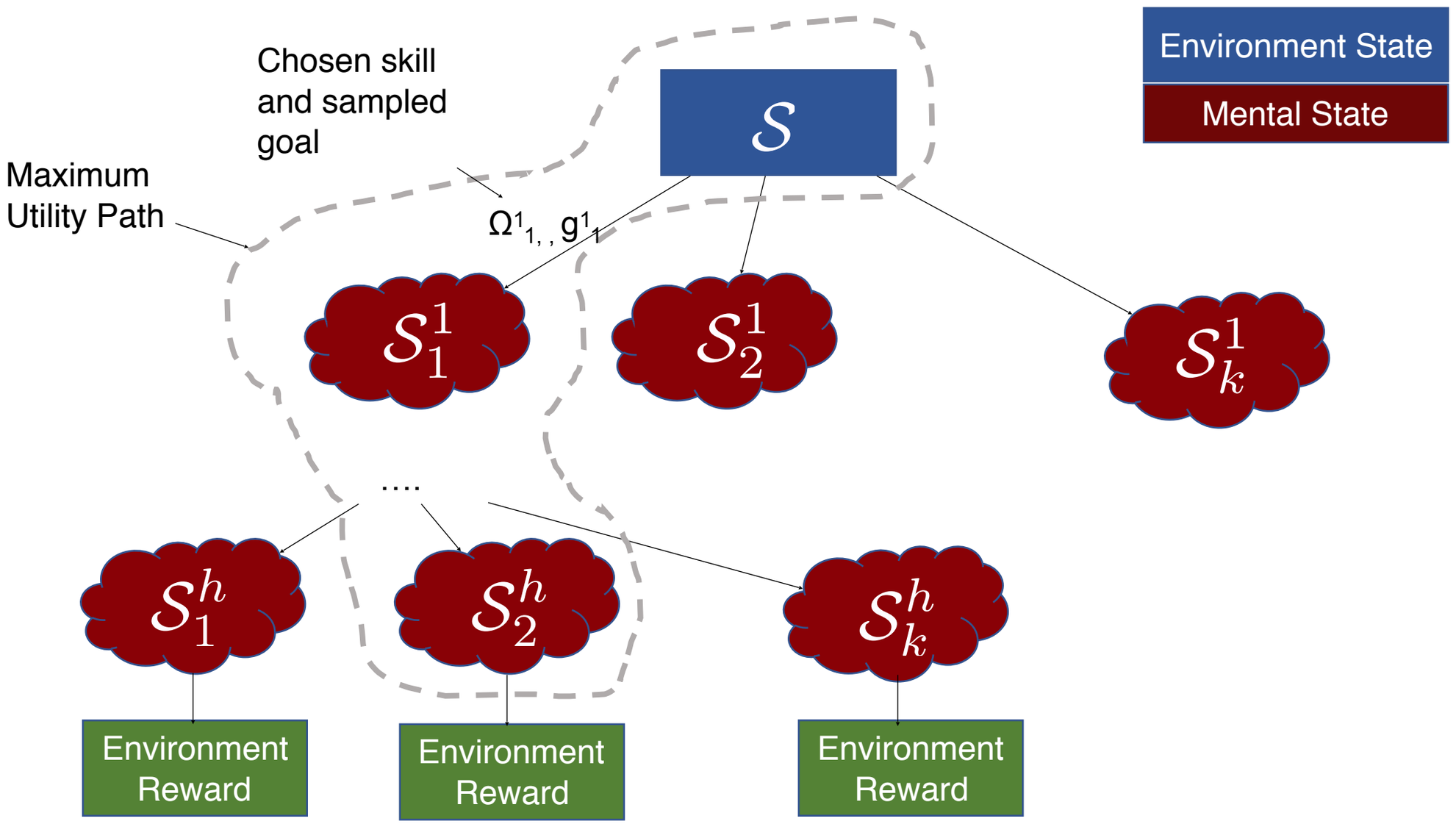
Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Planning using goal-conditioned policies



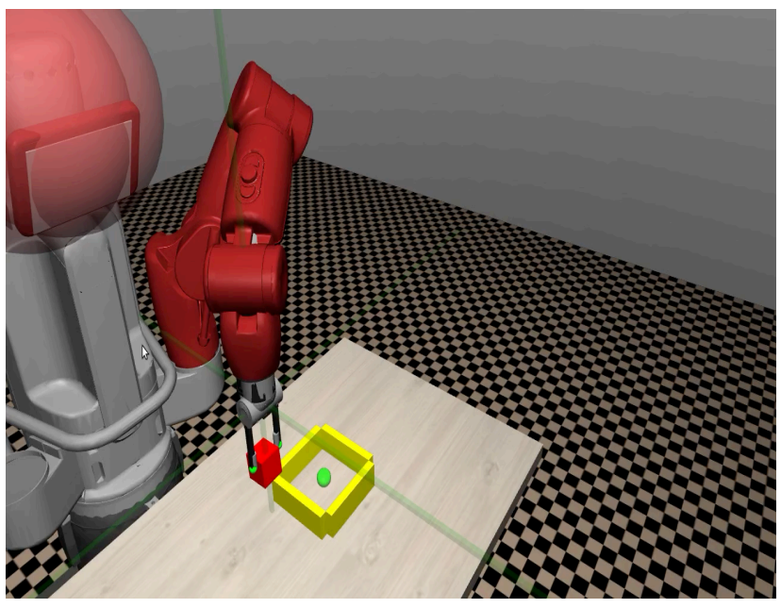
Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

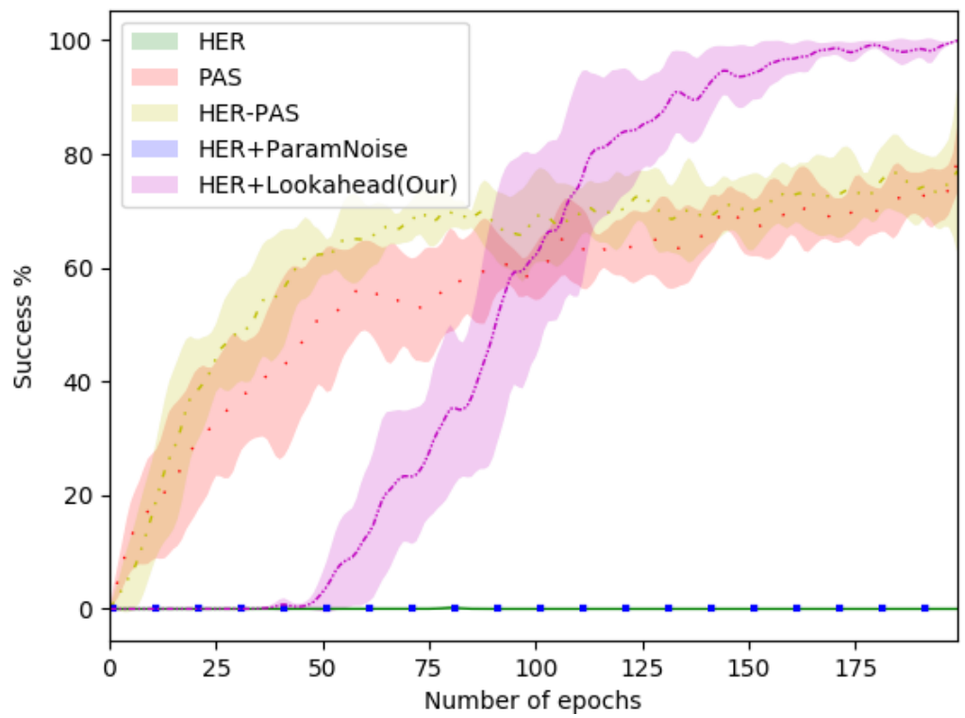
Planning using goal-conditioned policies



Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Planning using goal-conditioned policies





DDPG+HER

PAS MDP

PAS MDP + HER

HER + Parameter Noise

Look-Ahead exploration

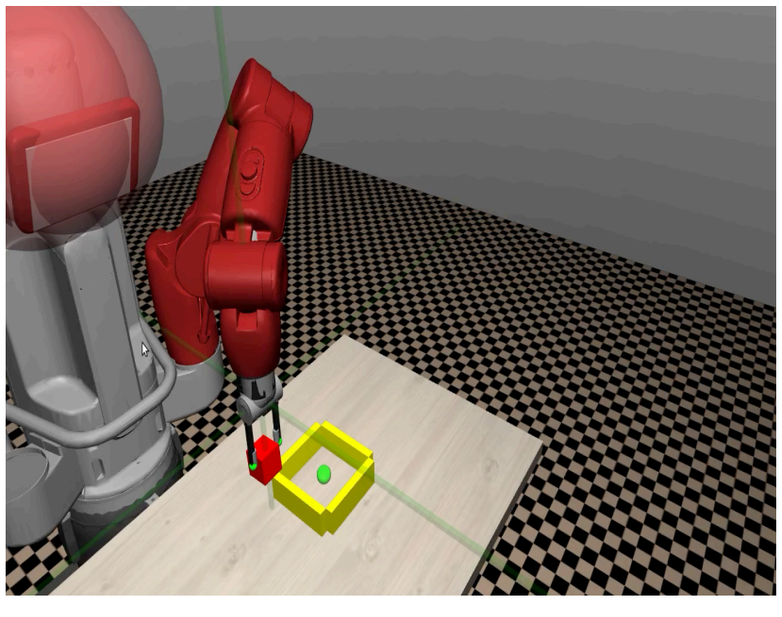
(c

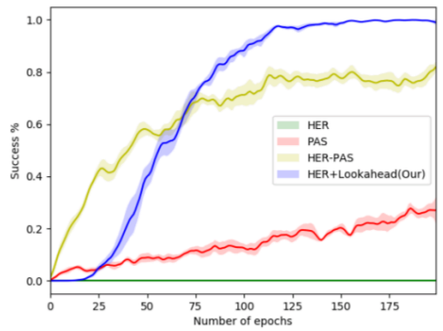
Where can you use model-based control?

During action selection at test time (online planning), or at training time, during exploration, or, to learn action selections and train a policy by imitation.

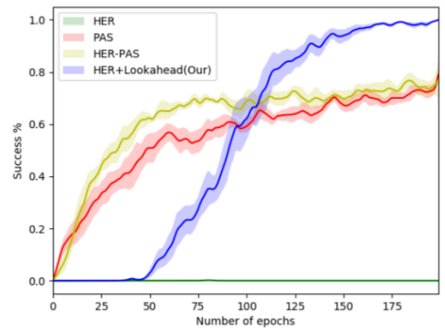
Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Planning using goal-conditioned policies

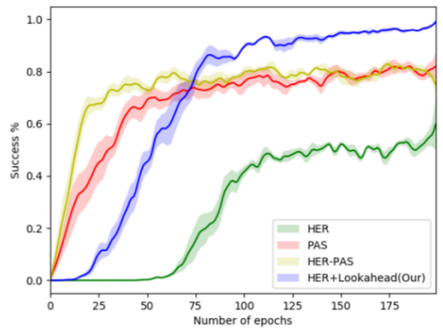




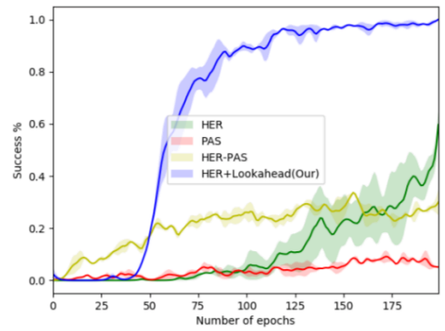
(a) Pick and Move



(b) Put A inside B



(c) Stack A on top of B

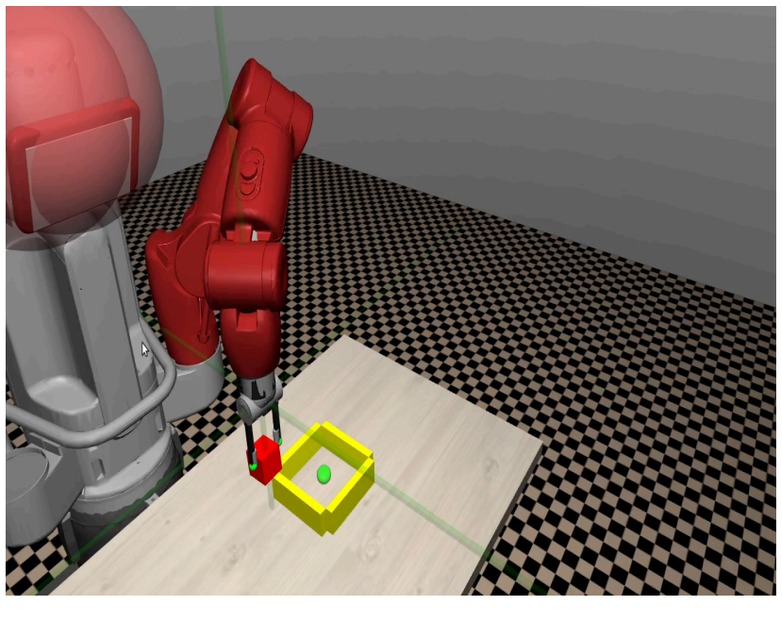


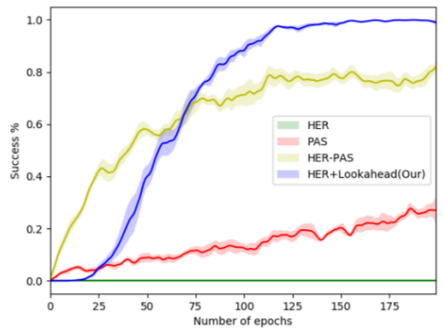
(d) Take A out of B

Here we use planning using goal-conditioned policies during exploration: if we fail to find a solution, we fall back to sampling random actions.

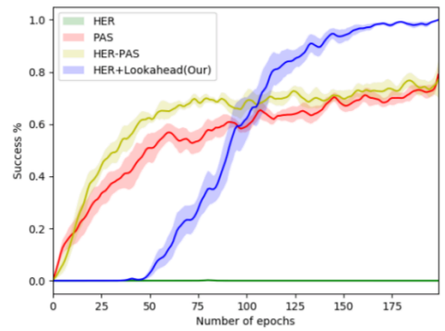
Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Planning using goal-conditioned policies

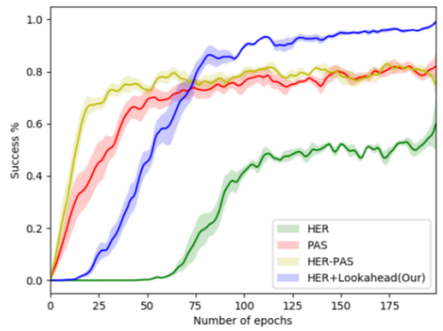




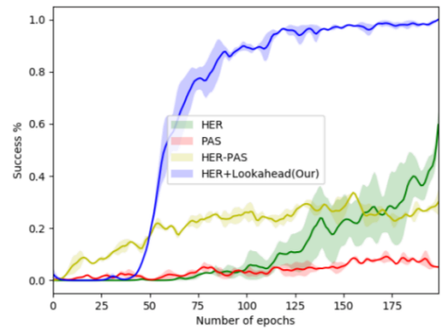
(a) Pick and Move



(b) Put A inside B



(c) Stack A on top of B



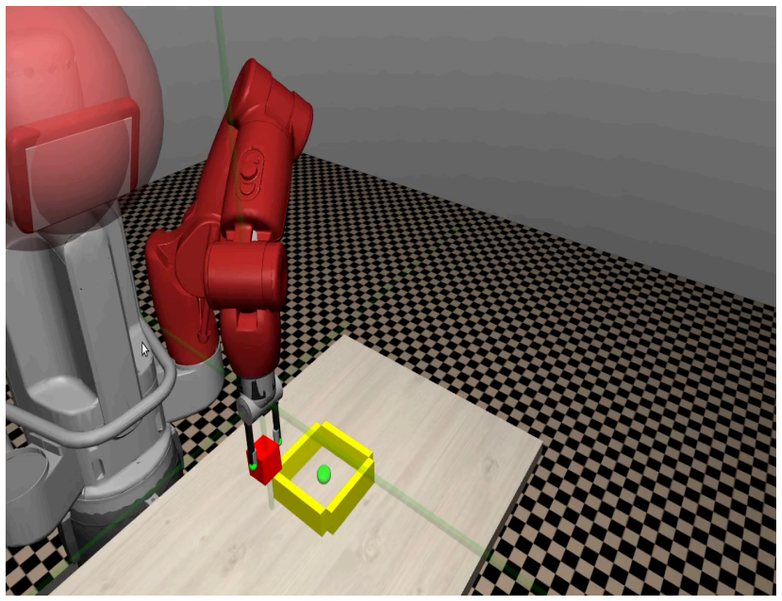
(d) Take A out of B

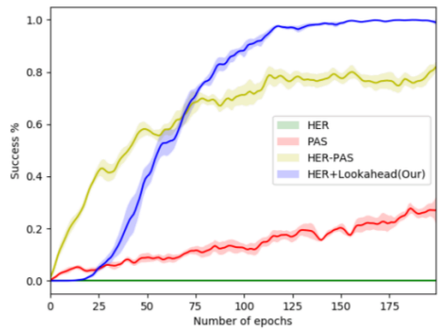
This was in a low-dim (engineered) state space.

- Does the state represent the yellow wall?
- What about obstacles at test time that are not part of the state at training time?
- What we would need to do to take this approach to sensory space?

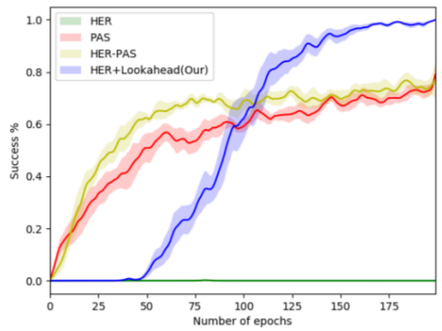
Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Planning using goal-conditioned policies

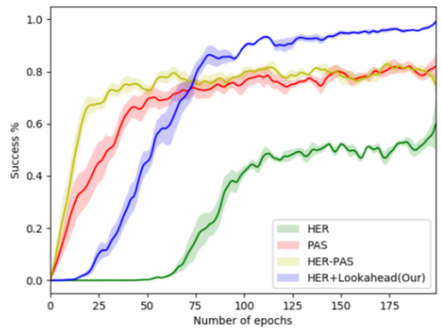




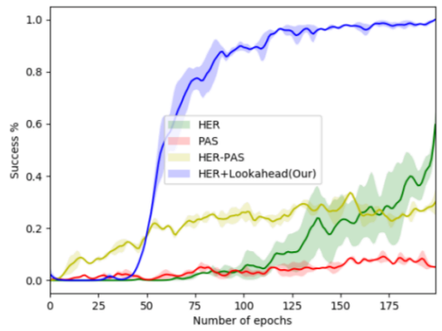
(a) Pick and Move



(b) Put A inside B



(c) Stack A on top of B



(d) Take A out of B

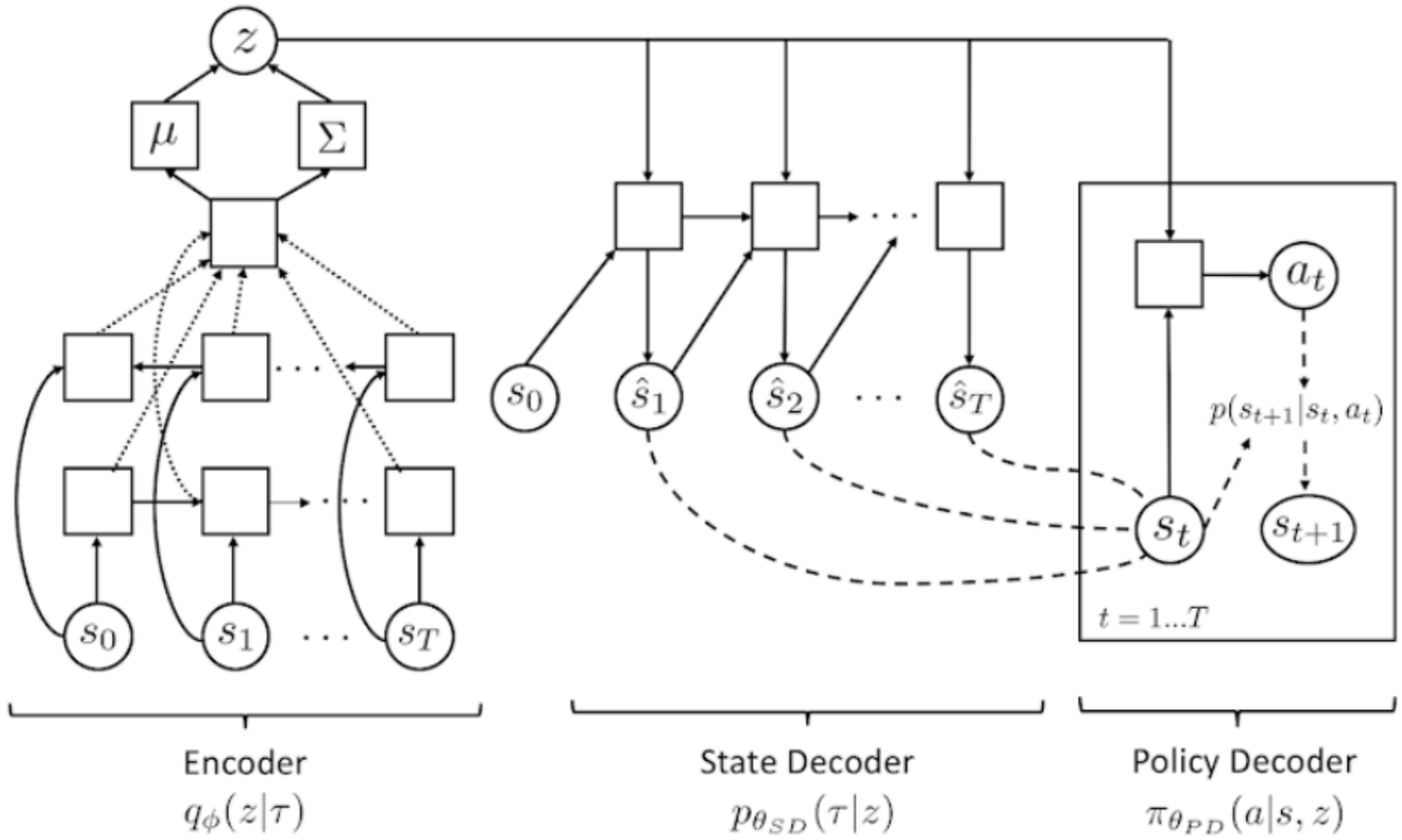
Since dealing with the sensory input is too hard, and is hard to get it to generalize, we will leave this problem to researchers and present another planning approach with temporal abstractions that solves:

- manual definition of the macro-actions
- conditioning on the plan as well, not just start state and goal state.

Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Self-Consistent Trajectory Autoencoder: Hierarchical Reinforcement Learning with Trajectory Embeddings

John D. Co-Reyes [*1] YuXuan Liu [*1] Abhishek Gupta [*1] Benjamin Eysenbach [2] Pieter Abbeel [1] Sergey Levine [1]



Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Self-Consistent Trajectory Autoencoder: Hierarchical Reinforcement Learning with Trajectory Embeddings

John D. Co-Reyes [*1] YuXuan Liu [*1] Abhishek Gupta [*1] Benjamin Eysenbach [2] Pieter Abbeel [1] Sergey Levine [1]

Two steps like before:

- Learning the primitives (options, macro-actions, skills)
- Sequence the primitives

Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Self-Consistent Trajectory Autoencoder: Hierarchical Reinforcement Learning with Trajectory Embeddings

John D. Co-Reyes [*1] YuXuan Liu [*1] Abhishek Gupta [*1] Benjamin Eysenbach [2] Pieter Abbeel [1] Sergey Levine [1]

A trajectory is encoded into a latent distribution.
The distribution is sampled from and directly decoded into a sequence of states using a state decoder.
used to condition a policy decoder which produces trajectories through sequential execution in the environment.

Two steps like before:

- Learning the primitives (options, macro-actions, skills)
- Sequence the primitives

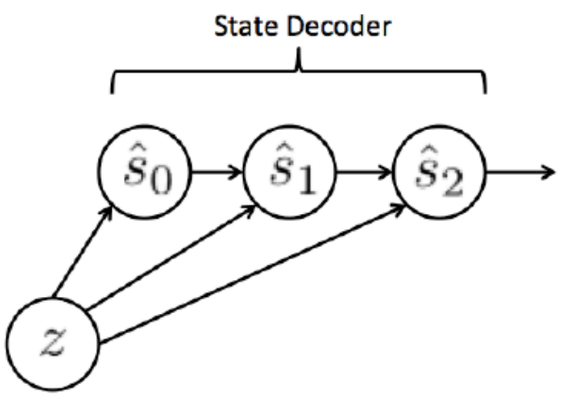
Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Learning the primitives: trajectories

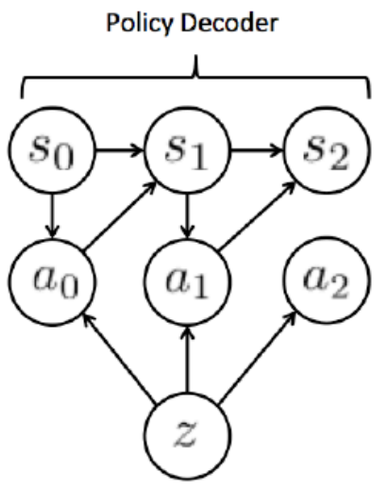
A trajectory is encoded into a latent code z. I sample code z (from a set of Gaussians zero mean and unit variance) and decode it into:

- a sequence of states in an autoregressive way: recursively predict a state, then feed it with the z and predict next s, and so on. It is an RNN decoder.
- a sequence of actions: I predict an action, i execute it in the environment, i get the next state, i feed it as input with the z, and predict next action, so on. It is a policy decoder.

This allow us to imagine the results of skills IF we would execute them, how would the world change as a result



This allow us to actually execute the skill in the world. Where is the goal?



Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Learning the primitives: trajectories

Q:How are we going to learn to embed trajectories to latent codes?

A:Variational auto-encoders!

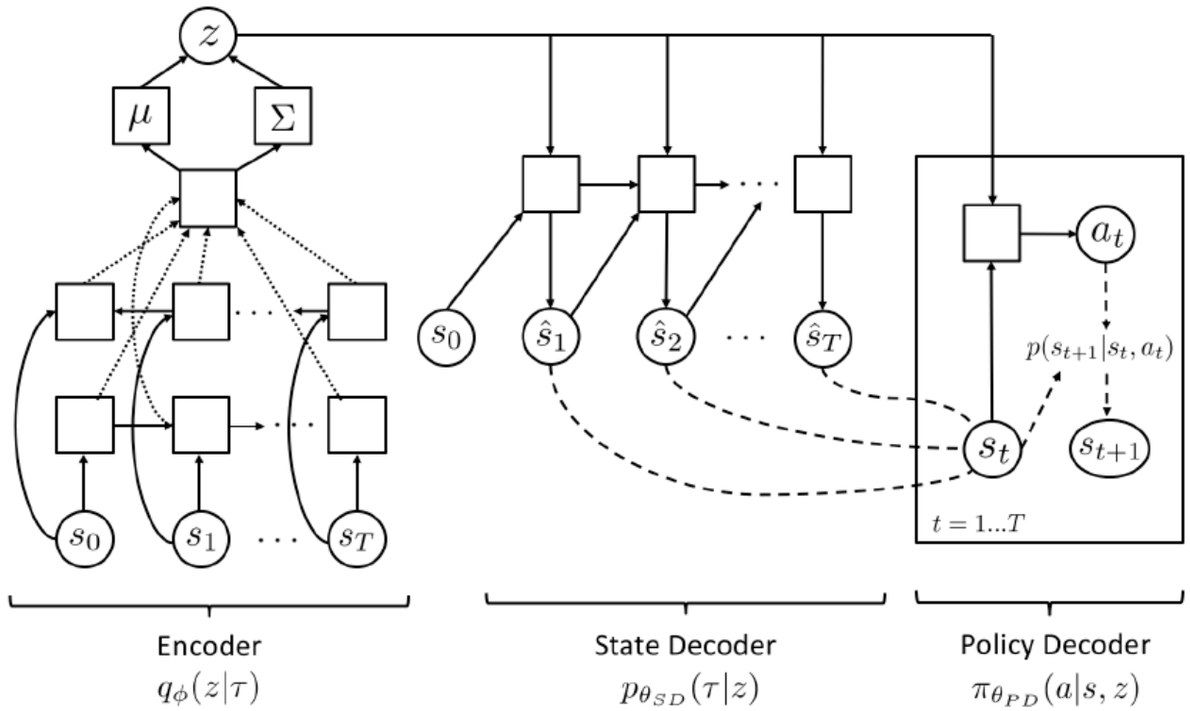
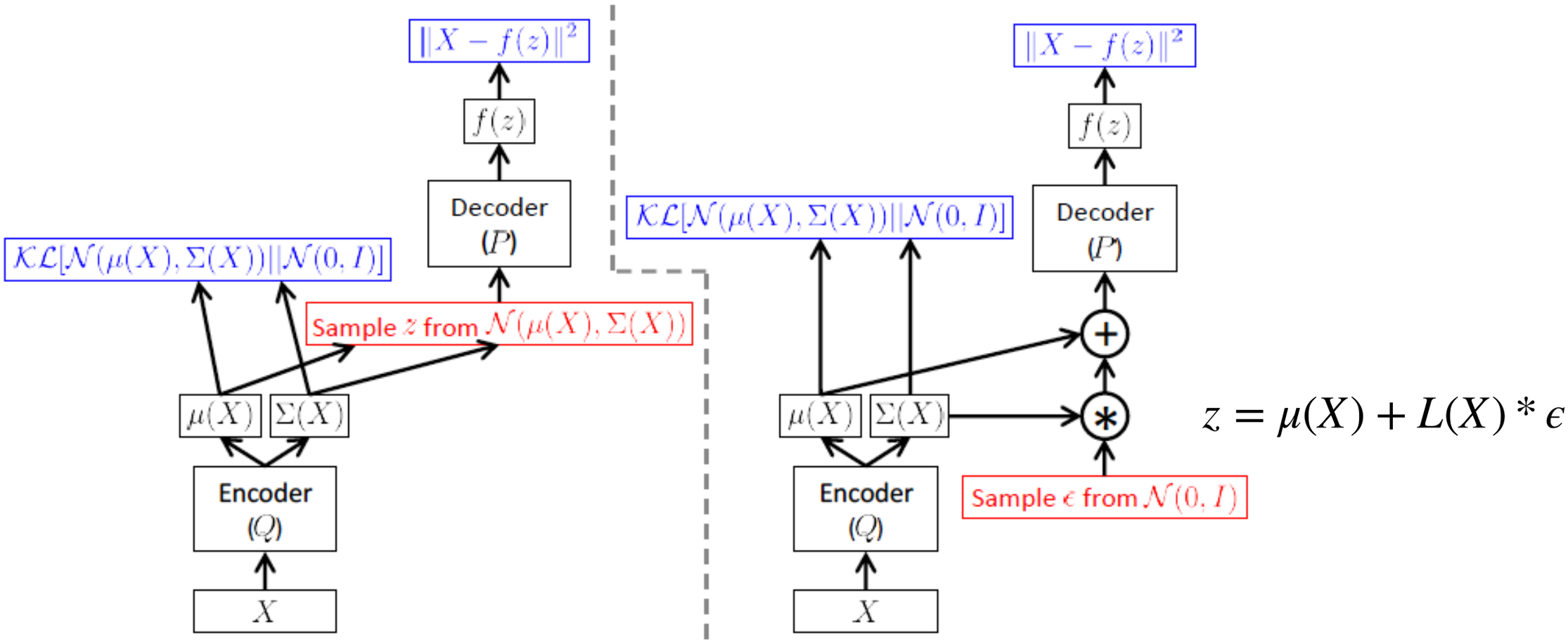


Figure 2. The SeCTAR model computation graph. A trajectory is encoded into a latent distribution, from which we sample a latent z. We then (1) directly decode z into a sequence of states using a recurrent state decoder and (2) condition a policy decoder on z to produce the same trajectory through sequential execution in the environment.

Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Variational auto encoder

From left to right: re-parametrization trick!

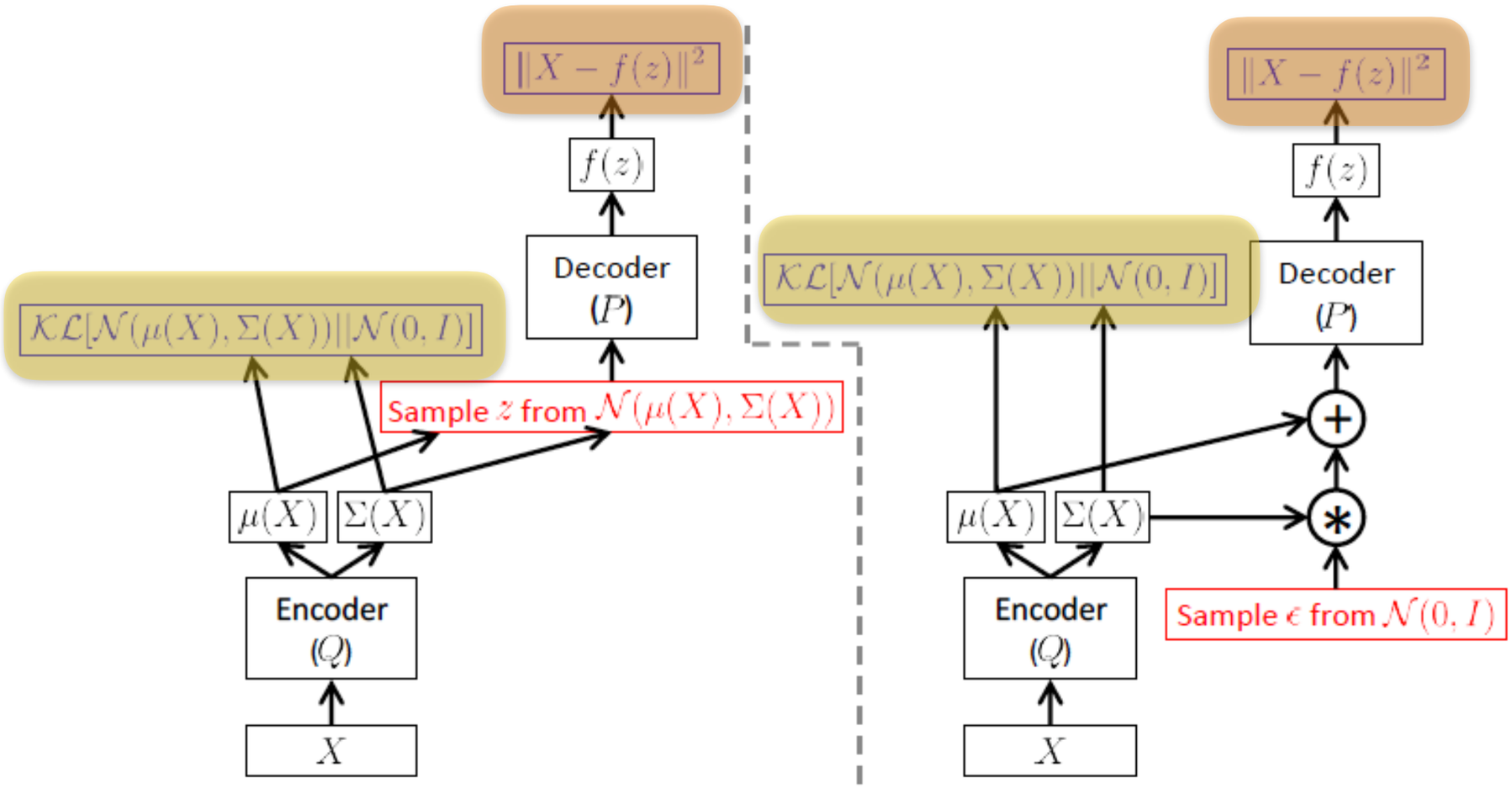


$$\min_{\phi,\theta} . \quad D_{KL}(Q(z|X;\phi)||P(z)) - \mathbb{E}_Q \log P(X|z;\theta)$$

Tutotial on variational Autoencoders, Doersch 2016

Variational auto encoder

From left to right: re-parametrization trick!



$$\min_{\phi,\theta} . \quad D_{KL}(Q(z|X;\phi)||P(z)) - \mathbb{E}_Q \log P(X|z;\theta)$$

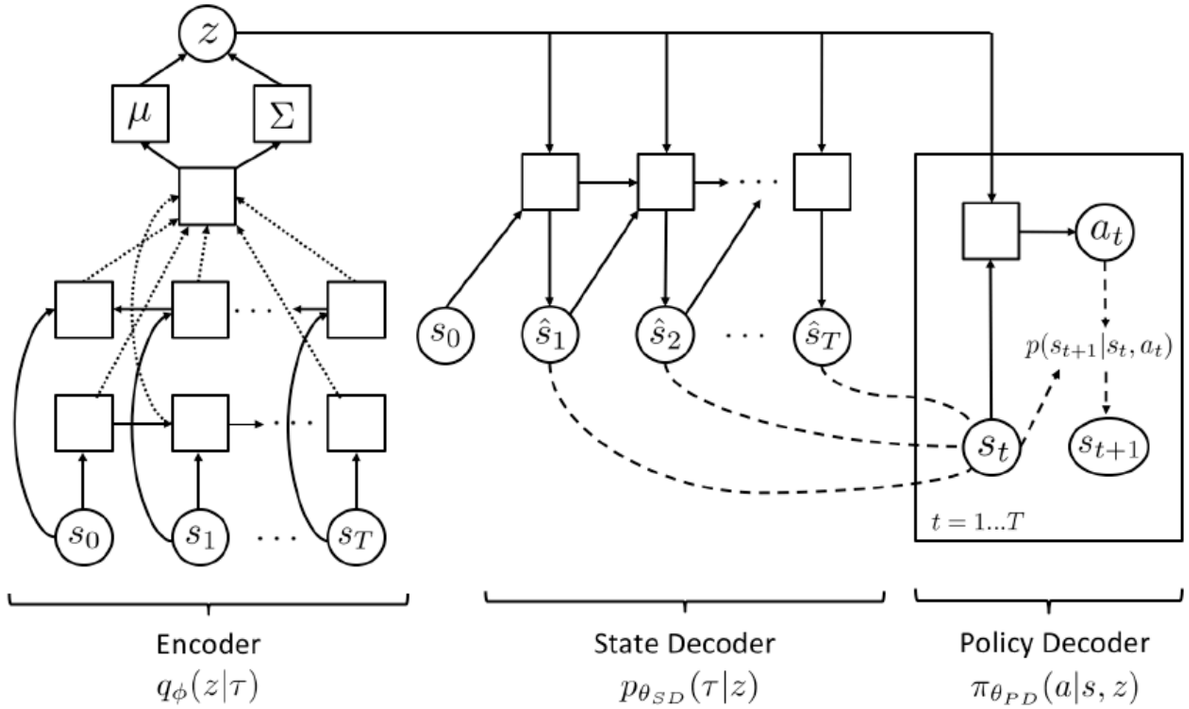
Tutotial on variational Autoencoders, Doersch 2016

Learning the primitives: trajectories

We just have two decoders to worry about..

The second decoder is a policy. How can we train it?

- Using behavior cloning,
- Using RL, where the reward is how well we hit the next state in the trajectory



Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Sequencing the primitives over time

Q: Given a large set of skills (each latent code gives us one) and their state sequence imaginations, how can we learn to link them over time?

A: Model-predictive control!

$$\max_{z_0, z_1 \ldots, z_{H/T}} \sum_{t=0}^{H/T} \gamma^{tT} R(\tau_t)$$

$$\text{subject to} \quad \tau_t \sim p_{\theta_{SD}}(\tau \mid z_t, s_0^t)$$

$$s_0^0 = s_0, s_0^{t+1} = s_T^t$$

Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Sequencing the primitives over time

Q: Given a large set of skills (each latent code gives us one) and their state sequence imaginations, how can we learn to link them over time?

A: Model-predictive control!

I search over the latent code space, one for each trajectory

$$\max_{z_0, z_1 \ldots, z_{H/T}} \sum_{t=0}^{H/T} \gamma^{tT} R(\tau_t)$$

$$\text{subject to} \quad \tau_t \sim p_{\theta_{SD}}(\tau \mid z_t, s_0^t)$$

$$s_0^0 = s_0, s_0^{t+1} = s_T^t$$

Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Sequencing the primitives over time

Q: Given a large set of skills (each latent code gives us one) and their state sequence imaginations, how can we learn to link them over time?

A: Model-predictive control!

Decode the latent z into a state sequence

Q: are we acting in the world?

$$\max_{z_0, z_1 \ldots, z_{H/T}} \sum_{t=0}^{H/T} \gamma^{tT} R(\tau_t)$$

$$\text{subject to} \quad \tau_t \sim p_{\theta_{SD}}(\tau \mid z_t, s_0^t)$$

$$s_0^0 = s_0, s_0^{t+1} = s_T^t$$

Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Sequencing the primitives over time

Q: Given a large set of skills (each latent code gives us one) and their state sequence imaginations, how can we learn to link them over time?

A: Model-predictive control!

The first state of the skill at t+1 should match the last state of the skill at time t

$$\max_{z_0, z_1 \ldots, z_{H/T}} \sum_{t=0}^{H/T} \gamma^{tT} R(\tau_t)$$

$$\text{subject to} \quad \tau_t \sim p_{\theta_{SD}}(\tau \mid z_t, s_0^t)$$

$$s_0^0 = s_0, s_0^{t+1} = s_T^t$$

Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Sequencing the primitives over time

Q: Given a large set of skills (each latent code gives us one) and their state sequence imaginations, how can we learn to link them over time?

A: Model-predictive control!

reward of trajectories

$$\max_{z_0, z_1 \ldots, z_{H/T}} \sum_{t=0}^{H/T} \gamma^{tT} R(\tau_t)$$

$$\text{subject to} \quad \tau_t \sim p_{\theta_{SD}}(\tau \mid z_t, s_0^t)$$

$$s_0^0 = s_0, s_0^{t+1} = s_T^t$$

Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018

Sequencing the primitives over time

Algorithm 1 Model predictive control in latent space

Given: trained SeCTAR model, Reward function R
for timestep $t \in \{1, \ldots, H/T\}$ **do**
    Sample K sequences of latents from the prior where each sequence has H_{MPC} number of latents
    Use the state decoder to predict environment states of length $T \times H_{MPC}$ for each latent sequence.
    Evaluate the reward per sequence, and choose the best sequence of latents.
    Execute the policy decoder $\pi_{\theta_{PD}}(a \mid s, z)$ conditioned on the first latent z from the chosen sequence, for T steps starting at s_0^t.
end for

Skill-guided Look-ahead Exploration for Reinforcement Learning of Manipulation Policies, Agarwal et al. 1018