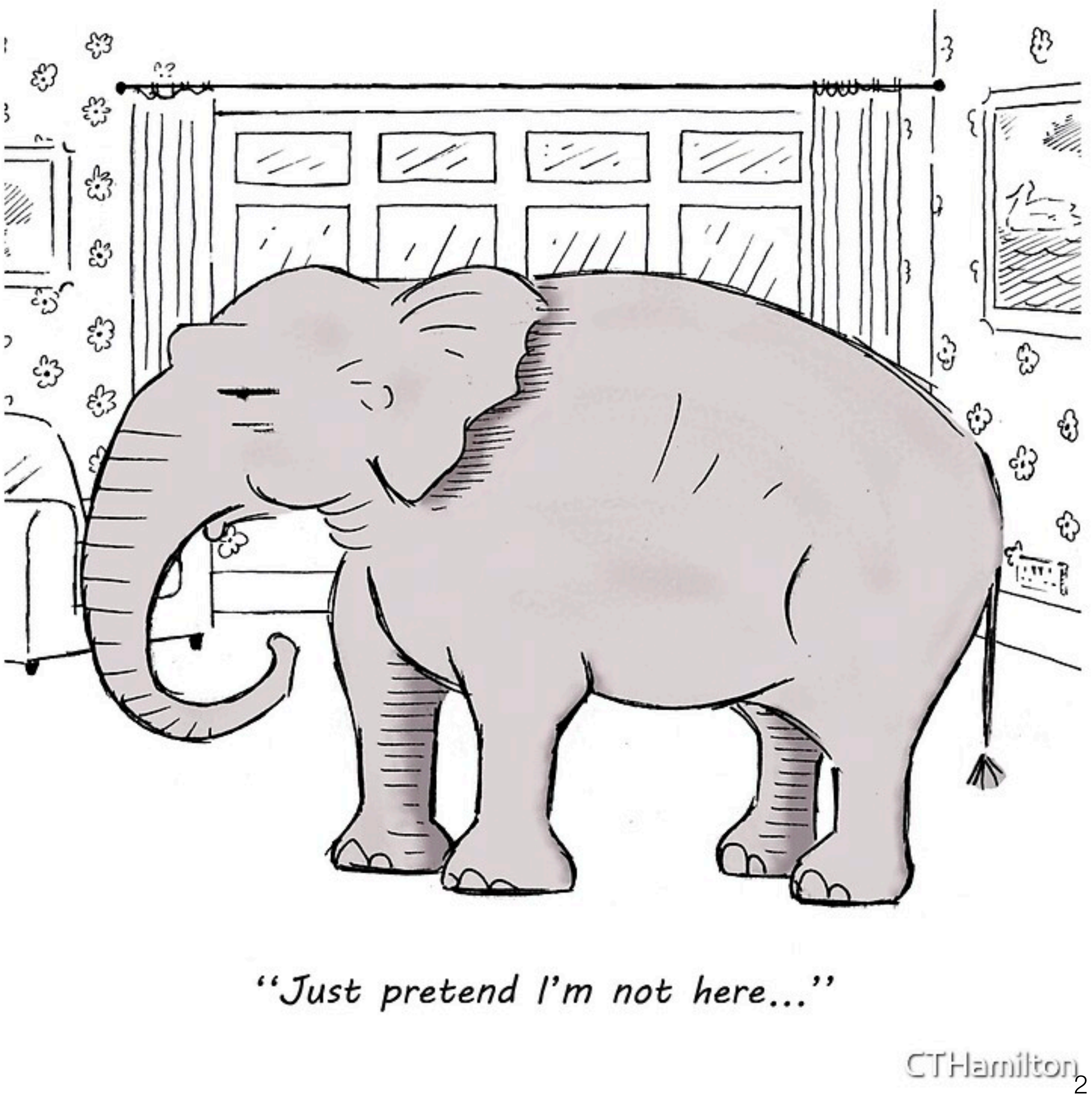
Planning with Inaccurate Models

Sanjiban Choudhury

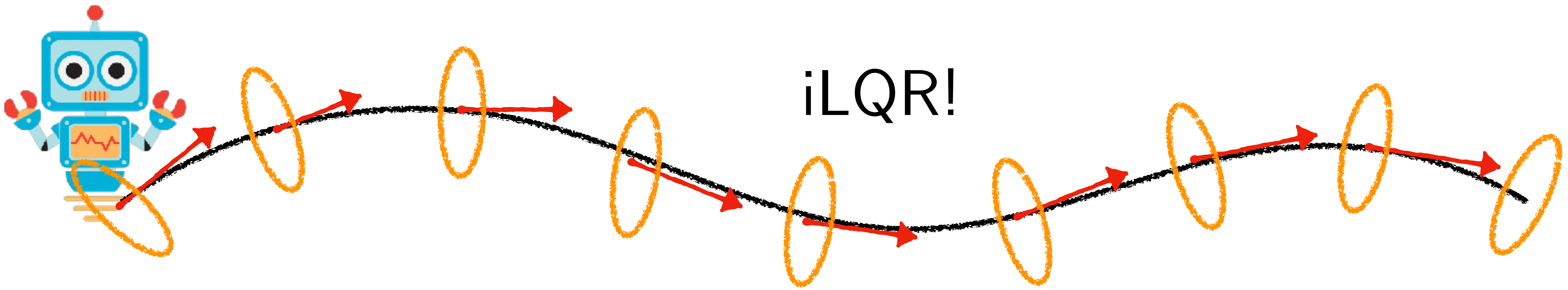
Cornell Bowers CIS
Computer Science

Elephant in the room: Why can't we just *learn a model*?



Model Based Reinforcement Learning

Learn Model → Plan with Learned Model



Why Model?

Models are *necessary*

Robots can't just try out random actions in the world!

Models are *necessary*

We invested heavily in simulators for helicopters and self-driving to verify behaviors before deployment

Models work in *theory*

Model-Based Reinforcement Learning with a Generative Model is Minimax Optimal

Alekh Agarwal
Microsoft
alekha@microsoft.com

Sham Kakade
University of Washington
sham@cs.washington.edu

Lin F. Yang
University of California, Los Angeles
linyang@ee.ucla.edu

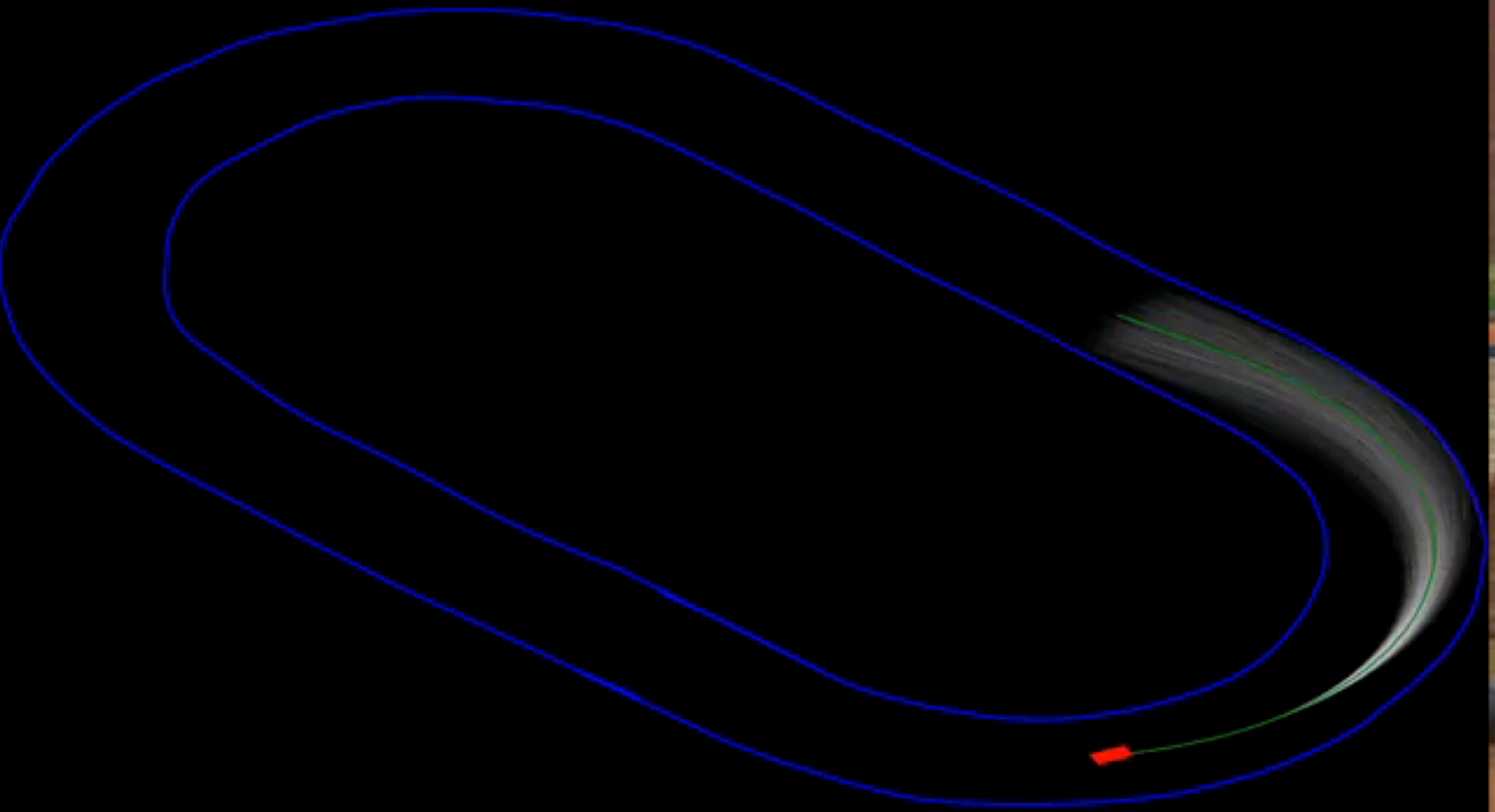
April 7, 2020

Models work in *practice*

Hafner et al. 2023

Learning Models.

2560, 2.5 second trajectories sampled
with cost-weighted average @ 60 Hz



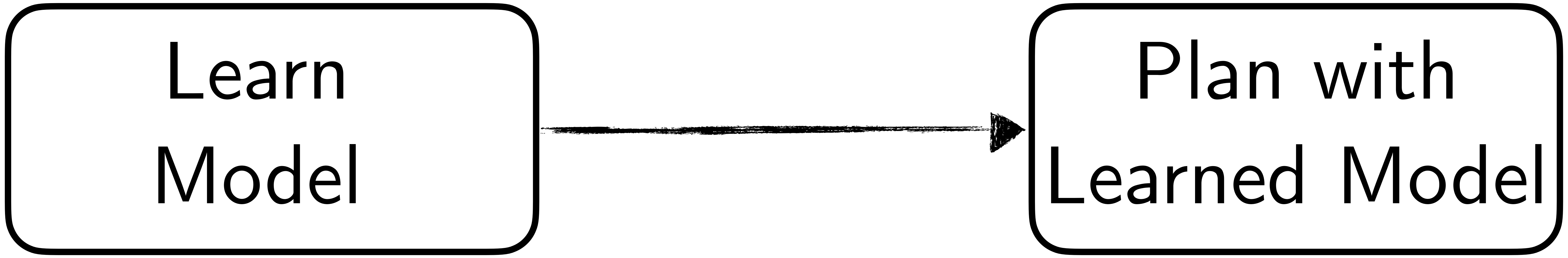
Georgia Tech Auto Rally (Byron Boots lab)

Activity!

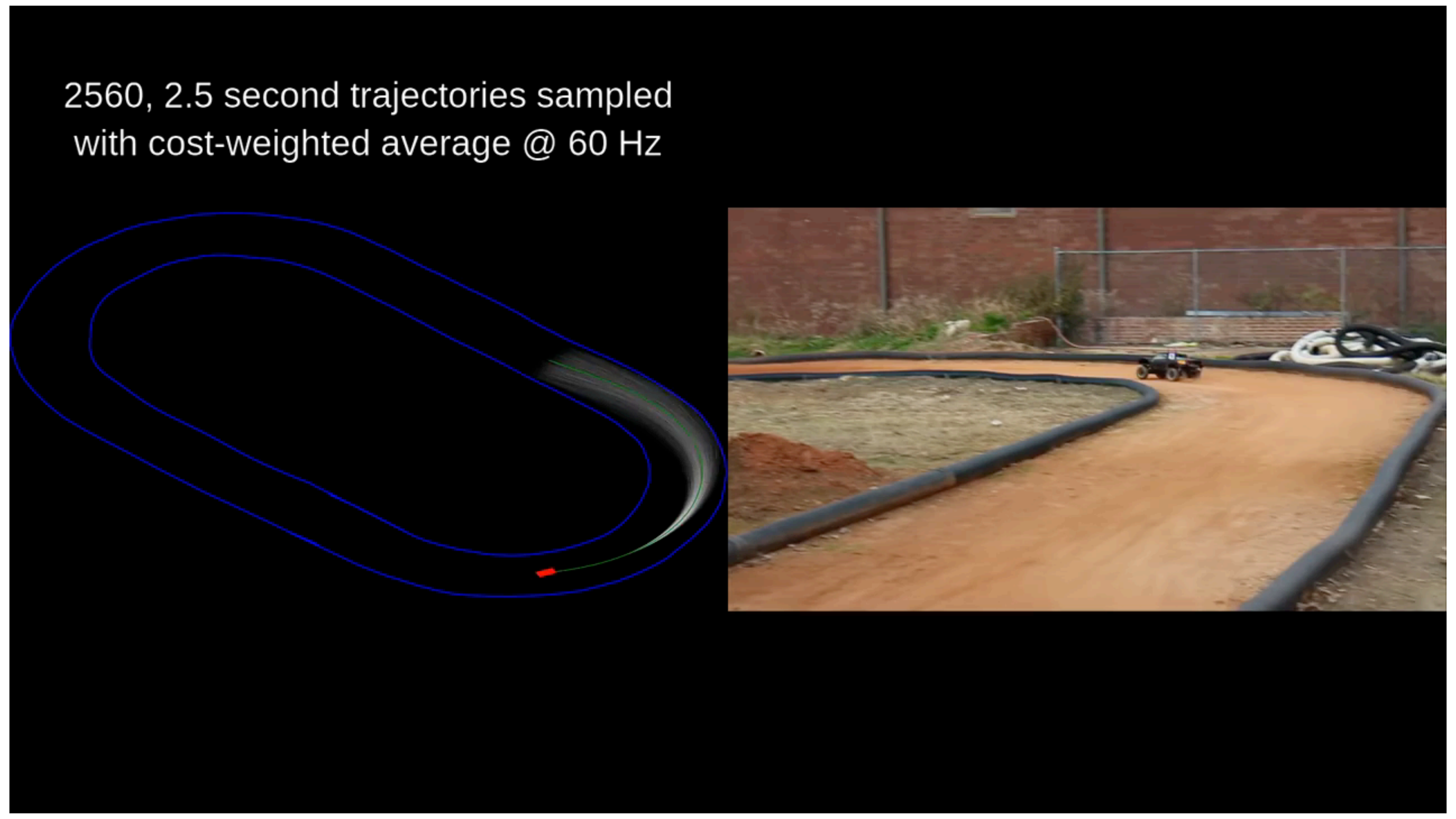
Think-Pair-Share

Think (30 sec): What features / architecture would you use to learn a model for rally car? What planner would you use?

Pair: Find a partner



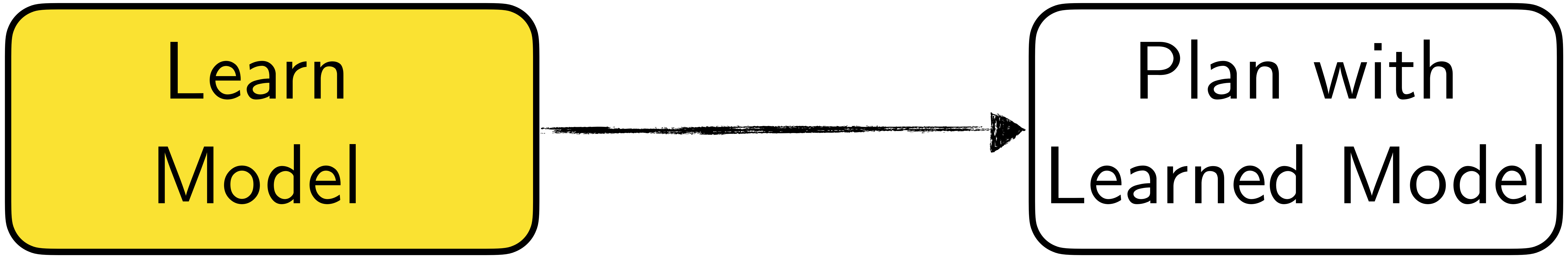
Share (45 sec): Partners exchange ideas



Part 1: System Identification

Information Theoretic MPC for Model-Based Reinforcement Learning

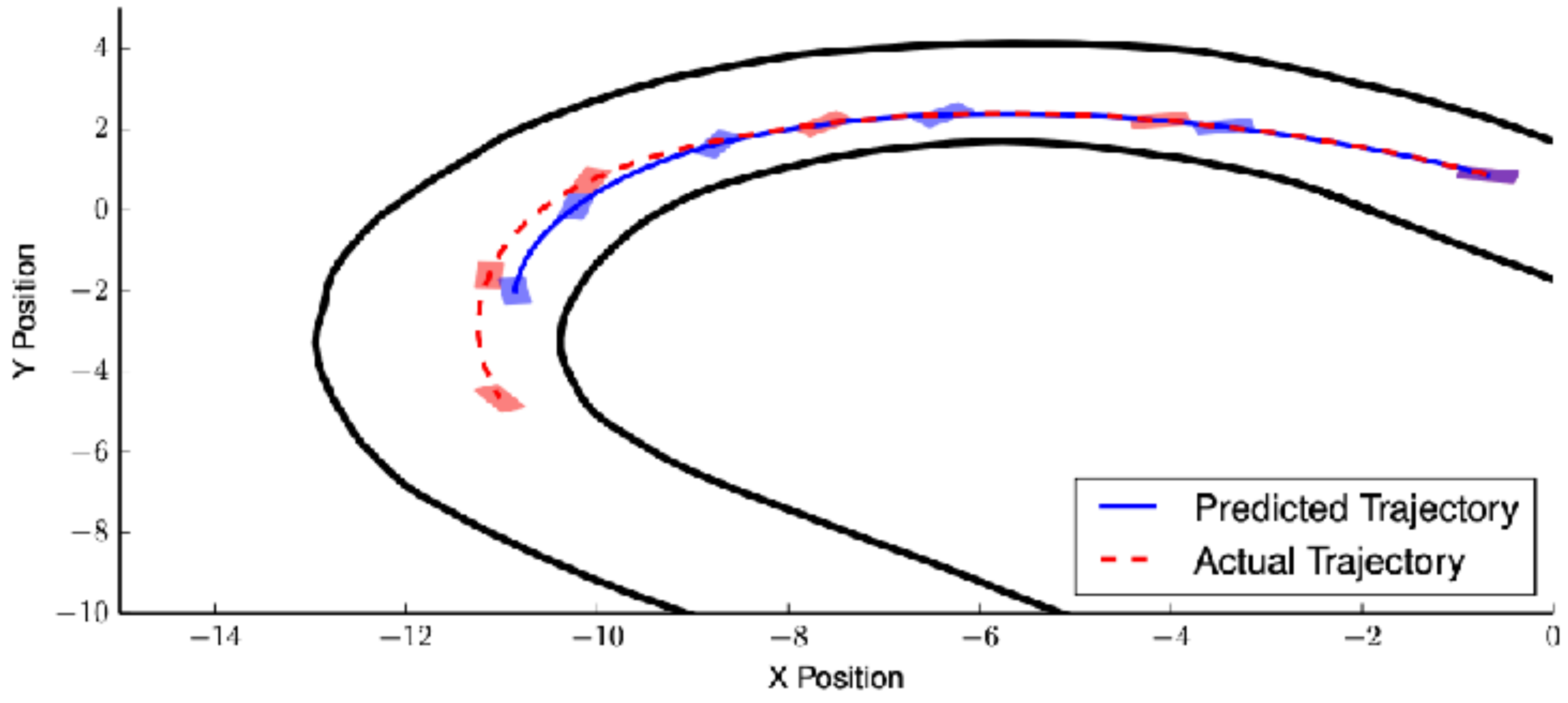
Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews,
James M. Rehg, Byron Boots, and Evangelos A. Theodorou



Collect data of rally car $(x_1, u_1, x_2, u_2, \ldots)$

$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t, \mathbf{u}_t) = \begin{bmatrix} \mathbf{q}_t + \dot{\mathbf{q}}_t \Delta t \\ \dot{\mathbf{q}}_t + \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)\Delta t \end{bmatrix}$$

2 Layer MLP



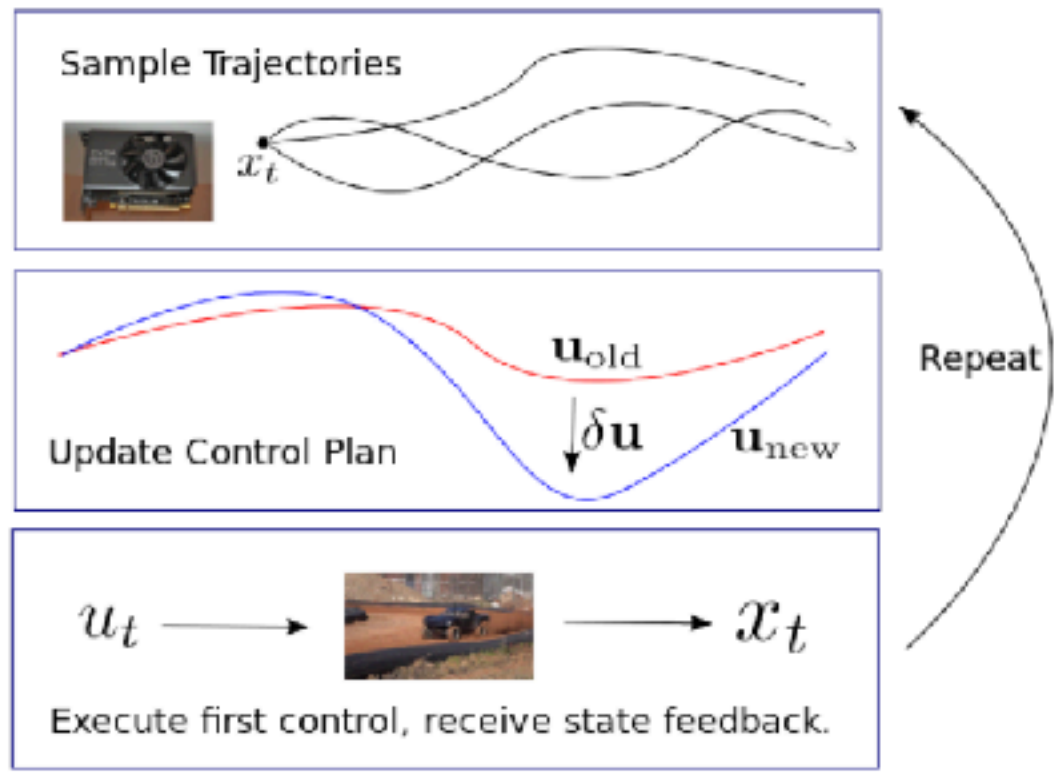
Part 2: Planning

Information Theoretic MPC for Model-Based Reinforcement Learning

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews,
James M. Rehg, Byron Boots, and Evangelos A. Theodorou

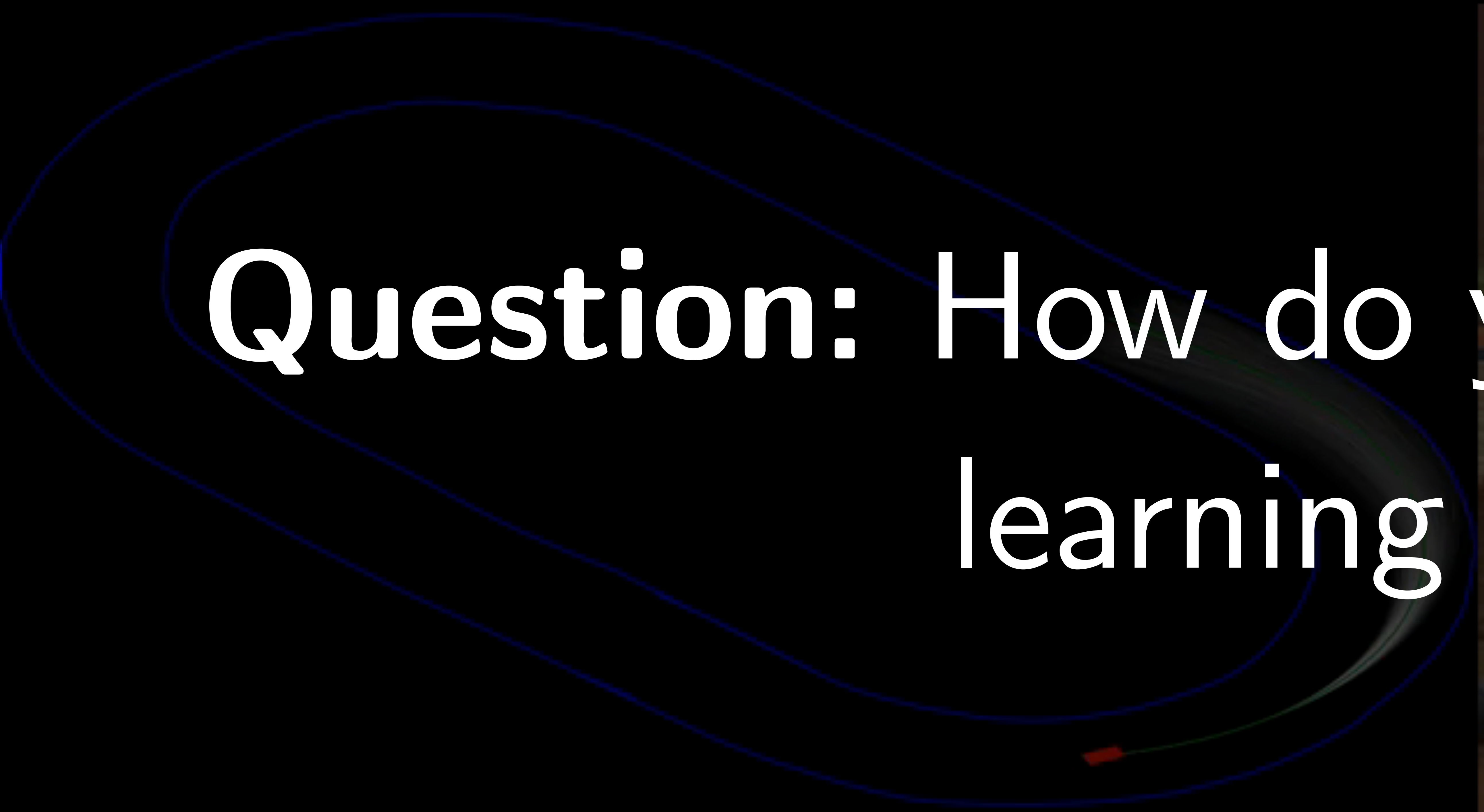
Learn Model → **Plan with Learned Model**

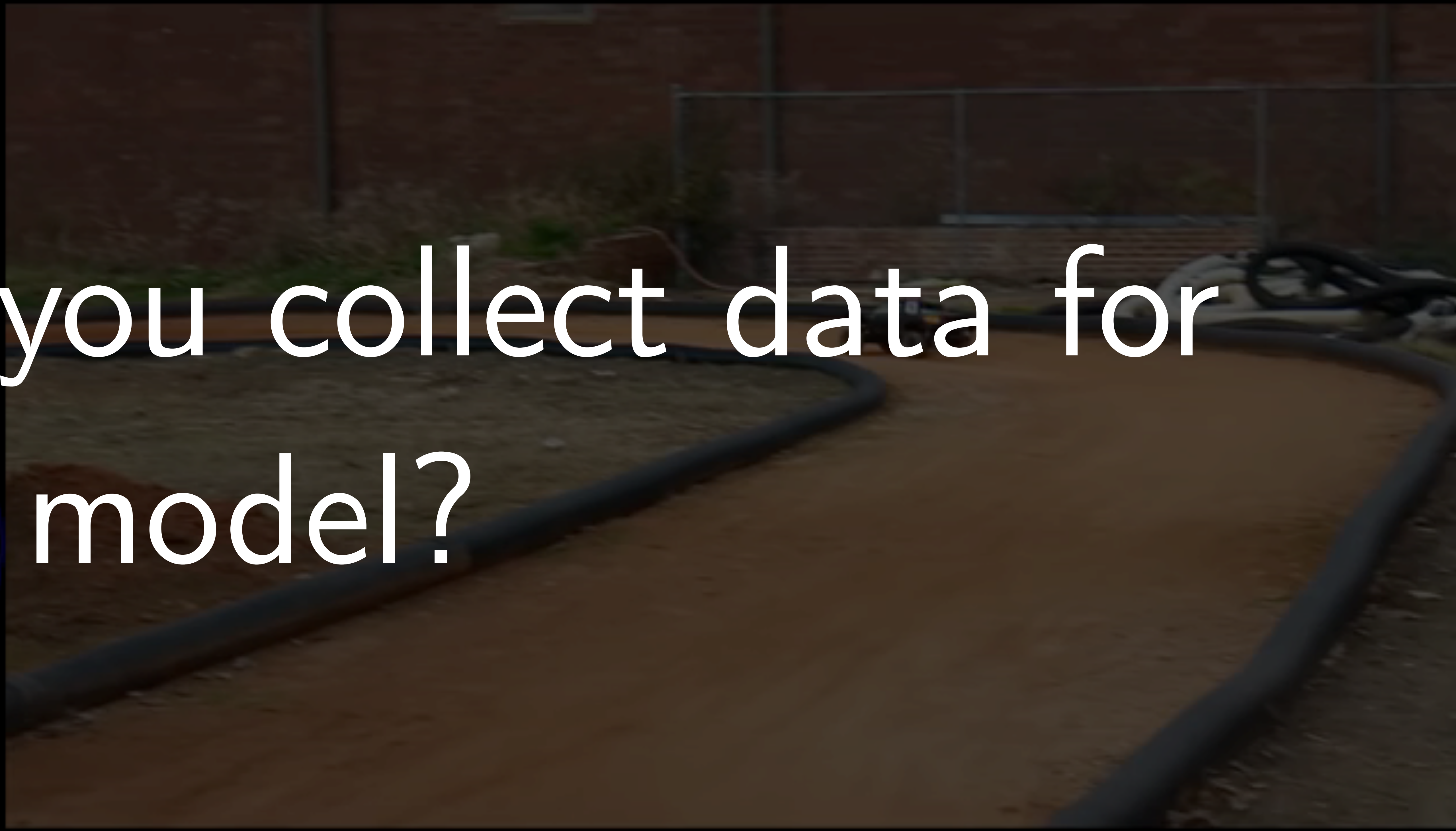
1. Sample and evaluate trajectories
2. Compute control update
3. Execute first control in sequence, receive state feedback
4. Repeat, using the un-executed portion of the previous control sequence to warm-start the trajectory



Cross Entropy like approach!

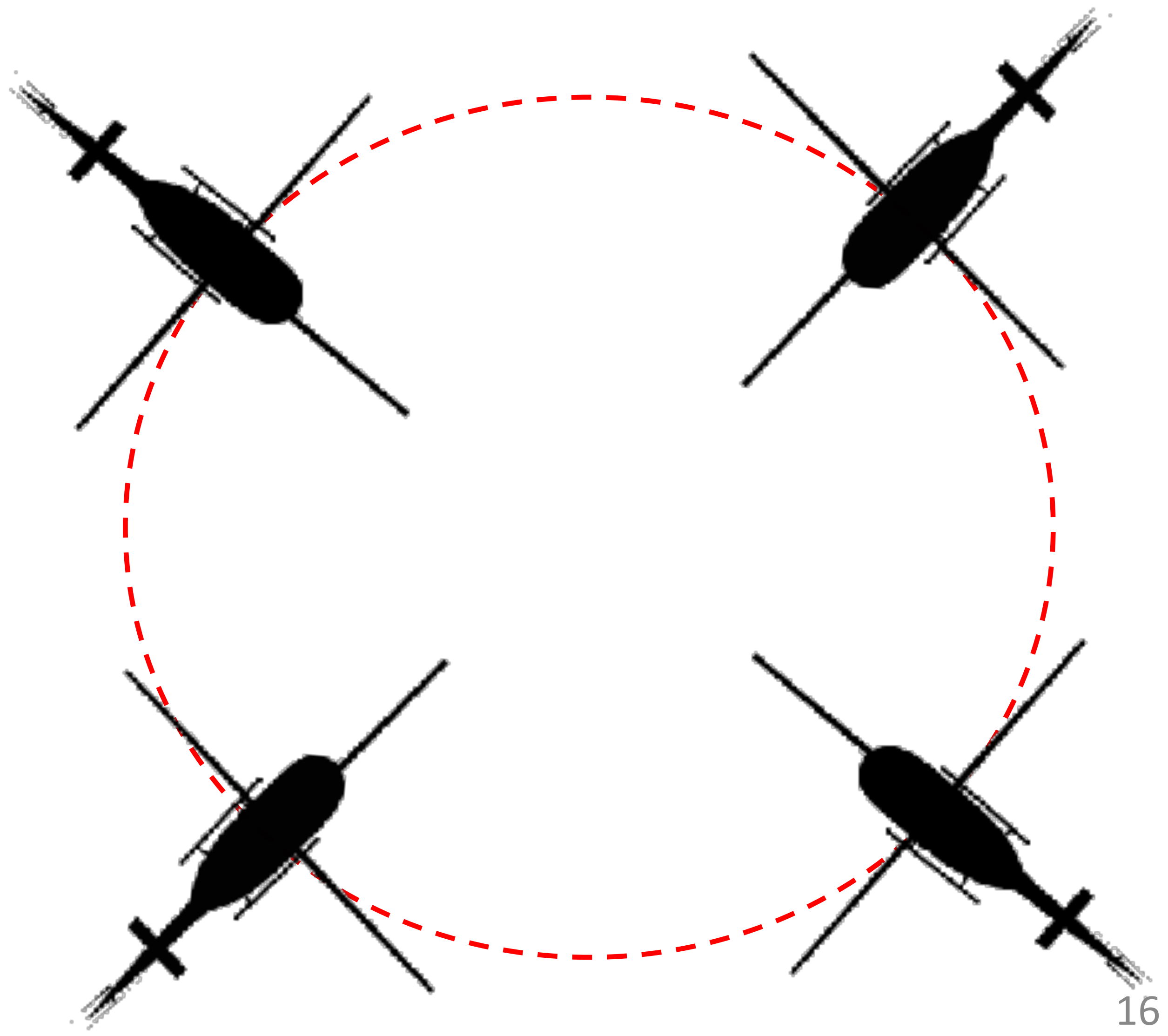
2560, 2.5 second trajectories sampled
with cost-weighted average @ 60 Hz





Question: How do you collect data for learning model?

Another Example: Helicopter Aerobatics

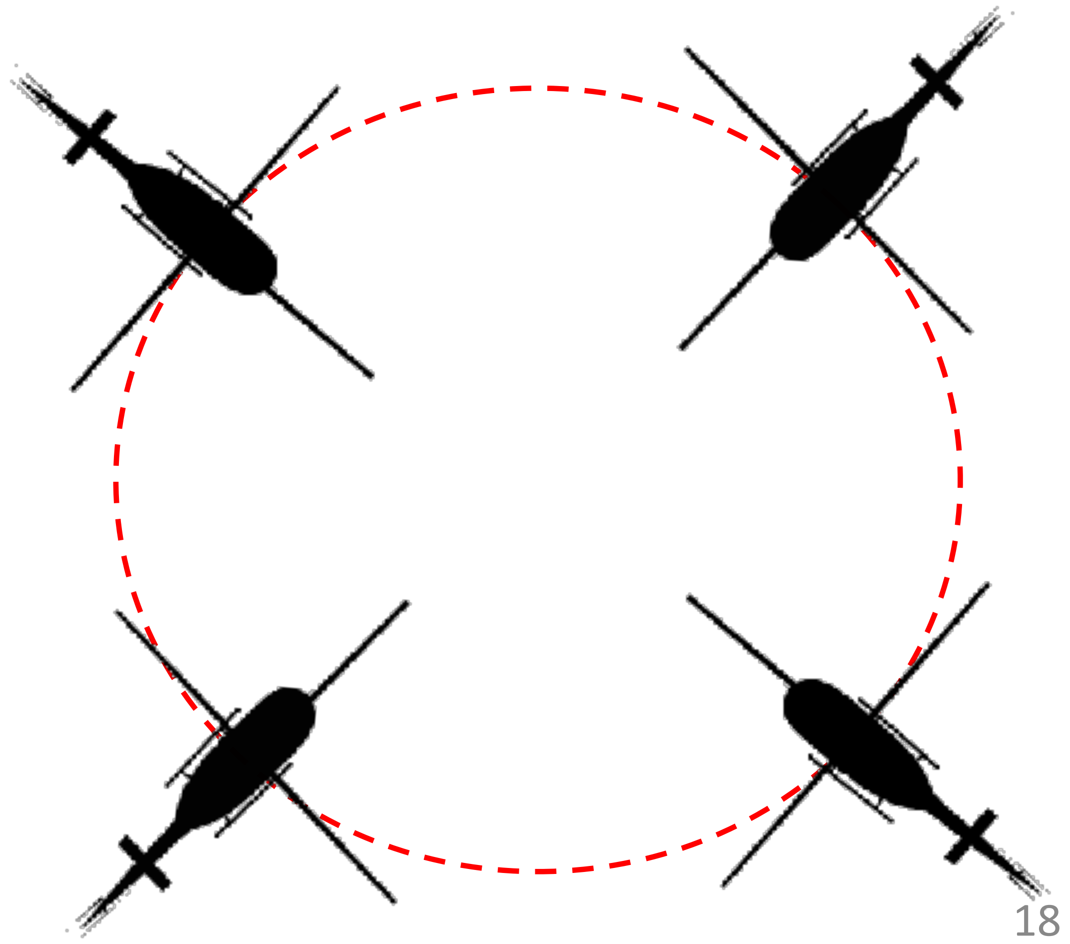


A nose-in funnel!

(Super cool work by Pieter Abeel et al. https://people.eecs.berkeley.edu/~pabbeel/autonomous_helicopter.html)

Part 1: System Identification

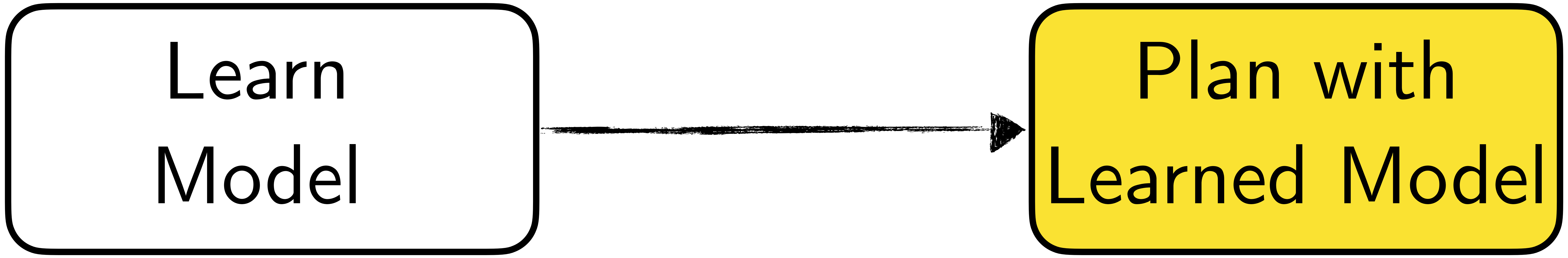
Learn Model → Plan with Learned Model

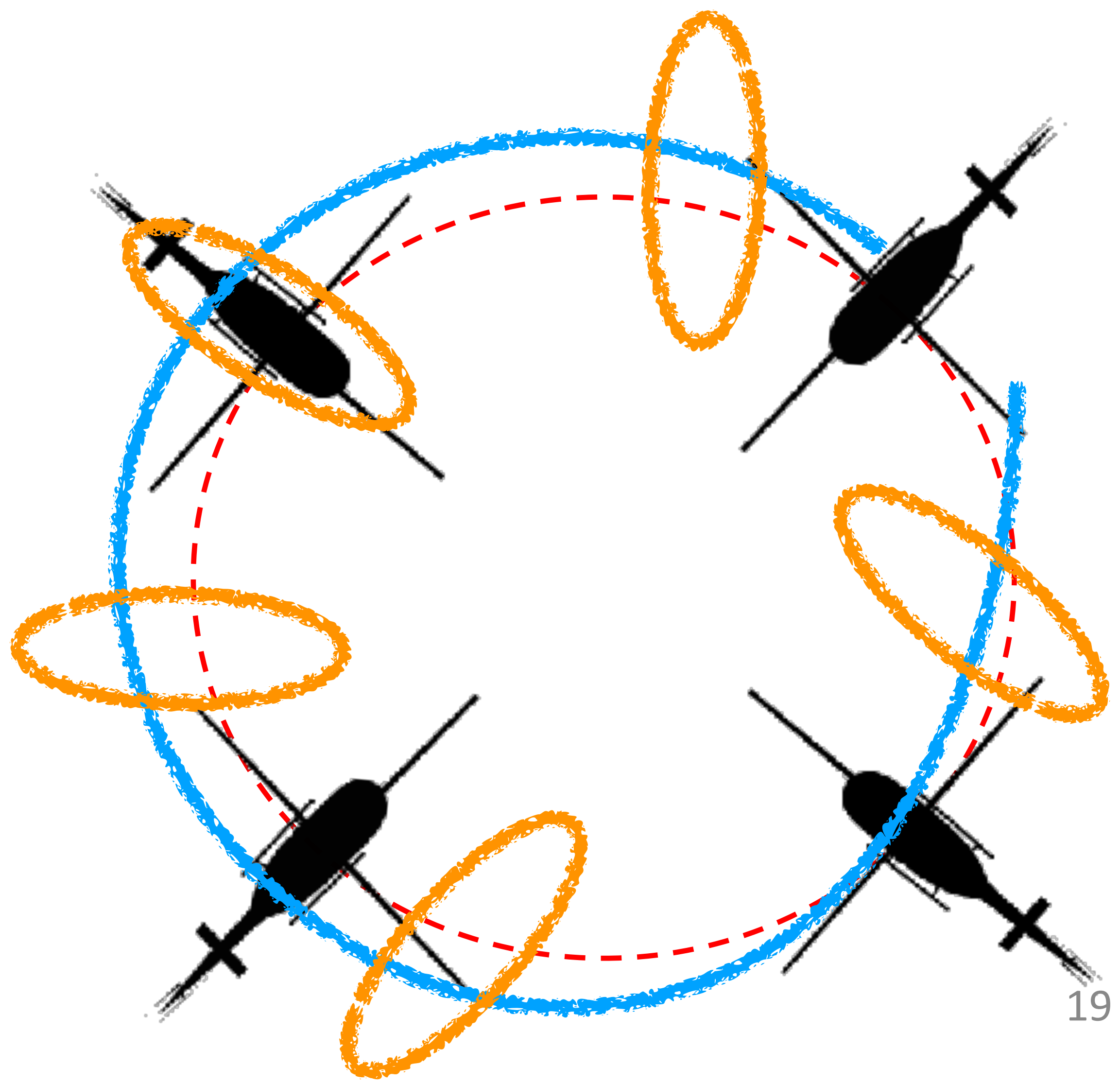


Learn a linear model around reference

$$\Delta x_{t+1} = A_t x_t + B_t u_t$$

Part 2: Planning





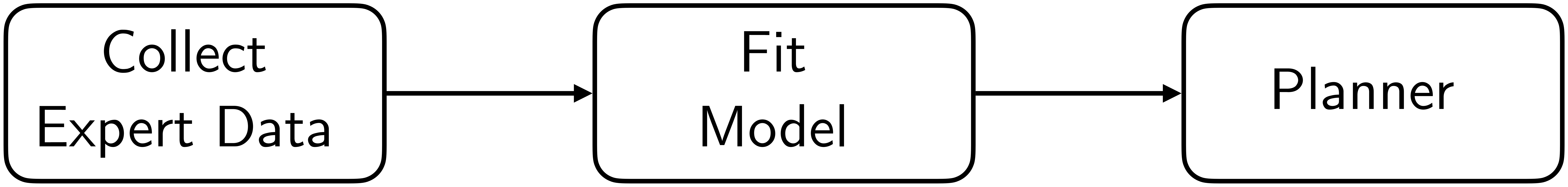
Use LQR with learnt models

How do we collect data
to train our model?

Strategy

Train a model on state actions visited by the expert!

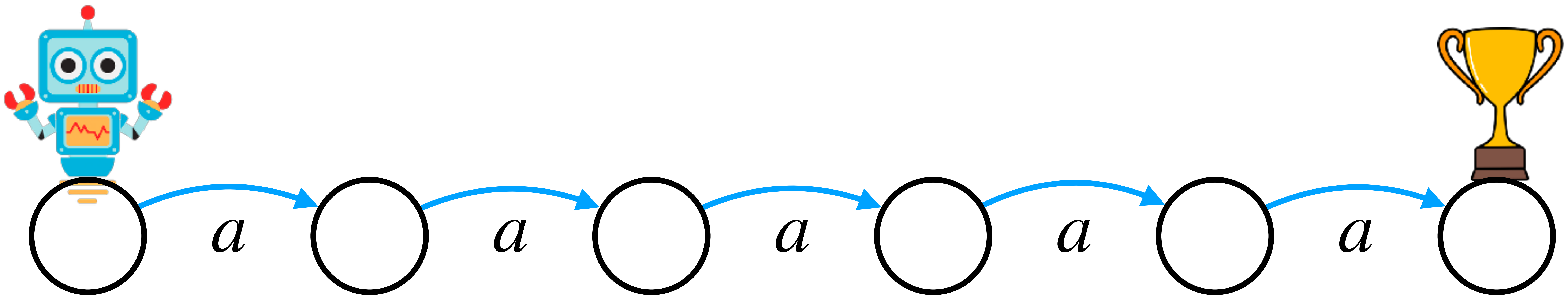
Model Based RL v1.0



*If I **perfectly** fit a model (i.e. training error zero),*
this should work, right?

World

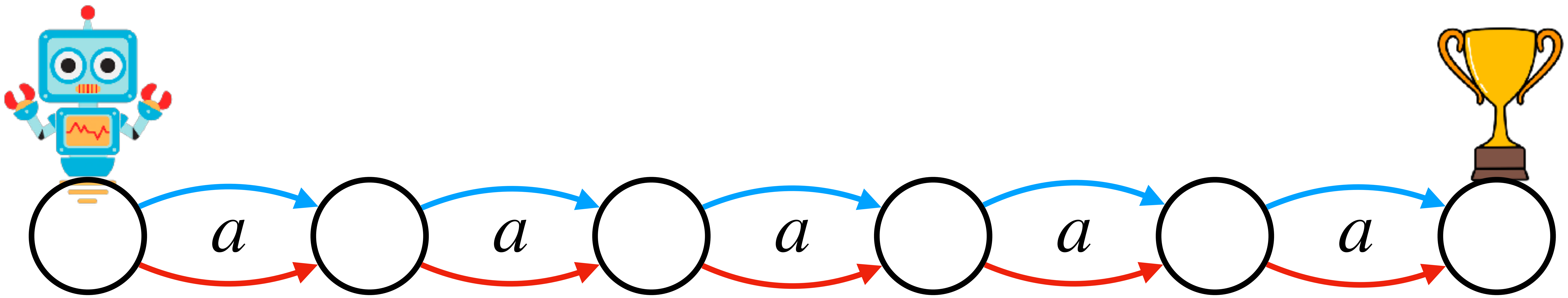
s'=$M^*(s, a)$



Experts picks action a to go to the goal

Model World

$s' = \hat{M}(s, a)$ $s' = M^*(s, a)$



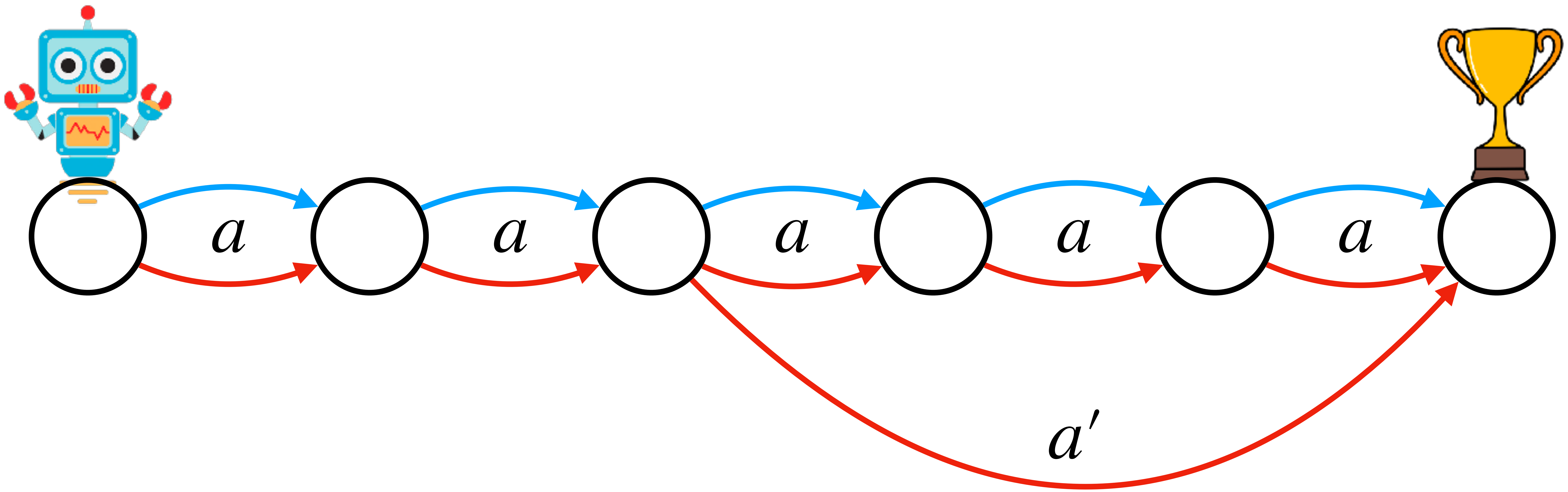
Model agrees with world, i.e. train error zero!

Model

s'$=\hat{M}(s,a)$

World

s'$=M^*(s,a)$



What if the model is optimistic?

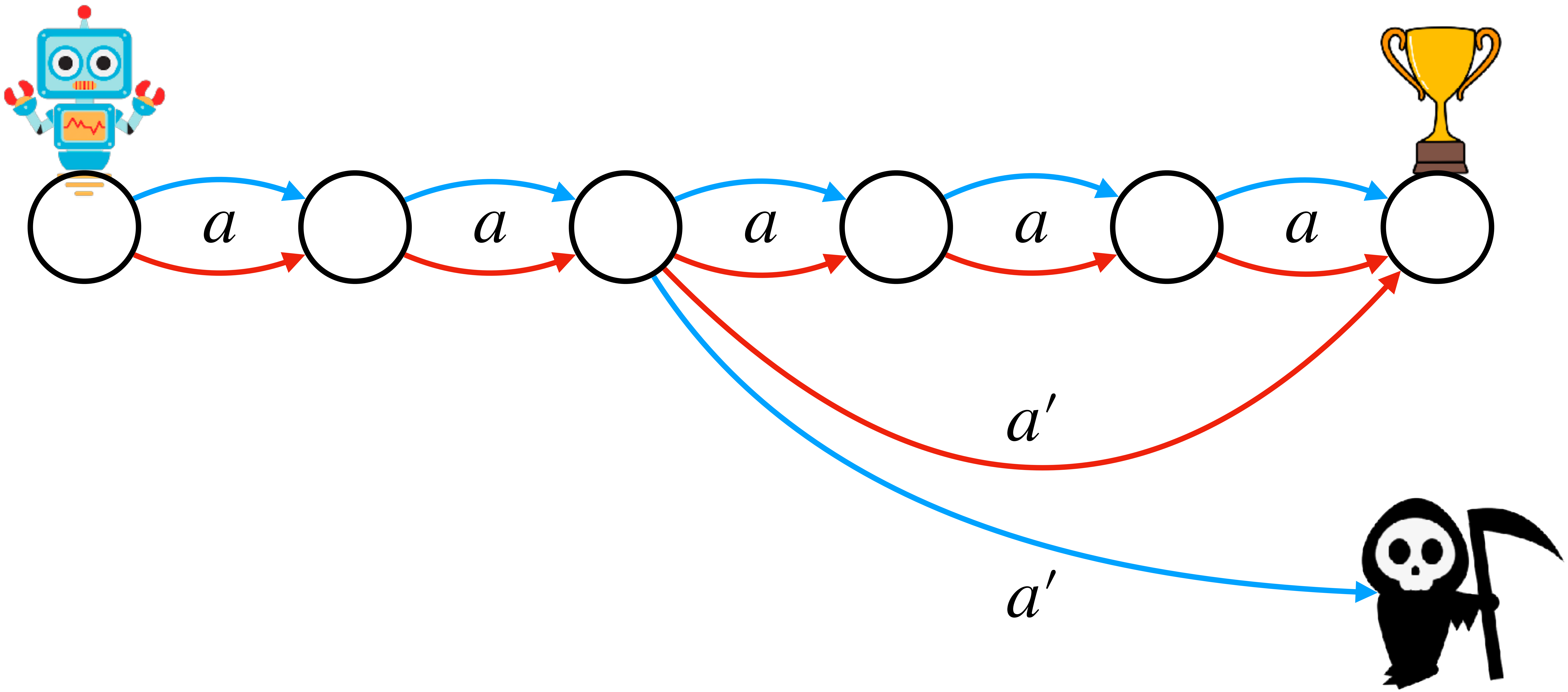
Predicts a short cut to the goal by taking action a'

Model

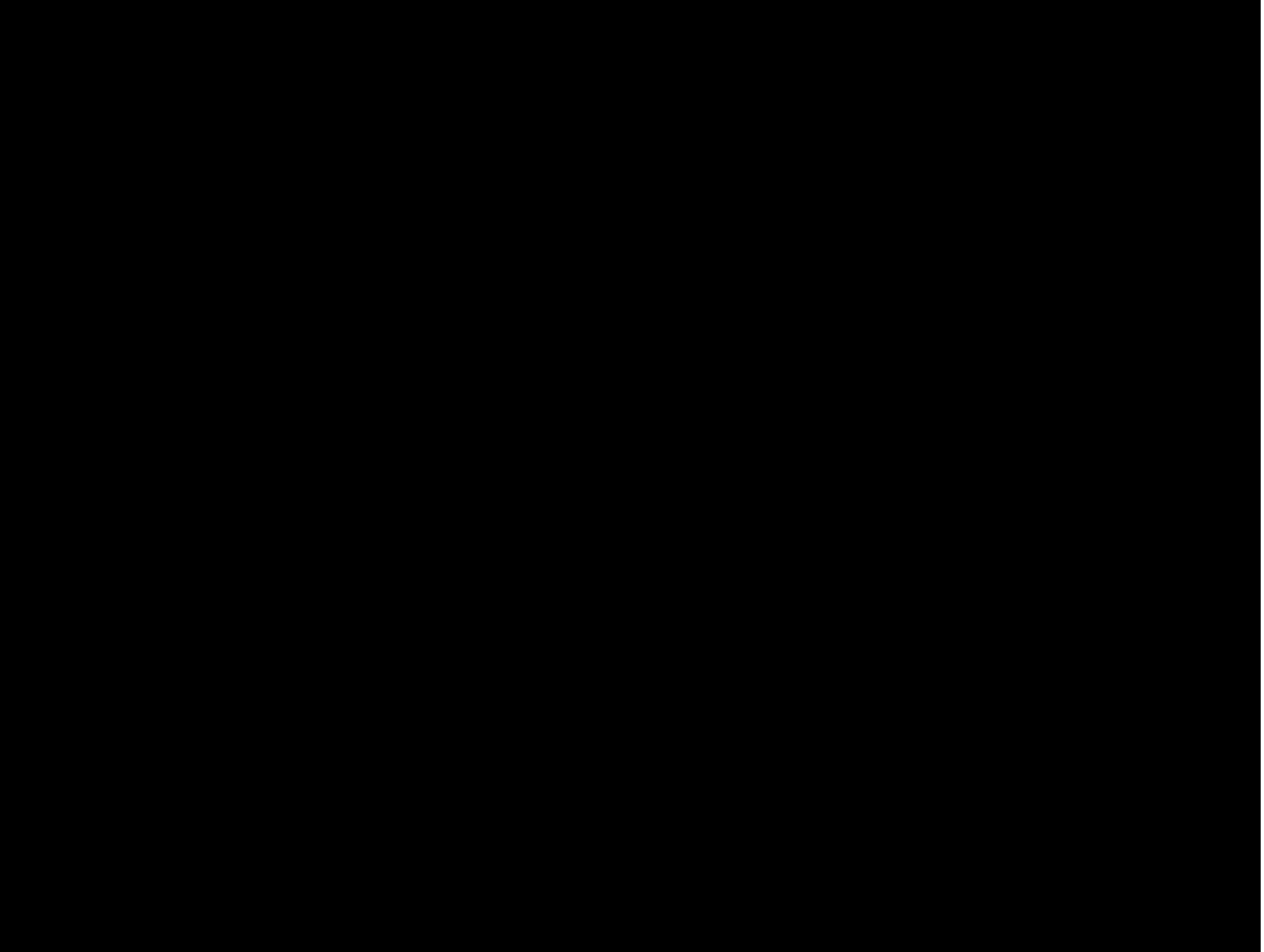
s'=$\hat{M}(s, a)$

World

s'=$M^*(s, a)$



In reality the shortcut ends in death …



Training on Expert Data

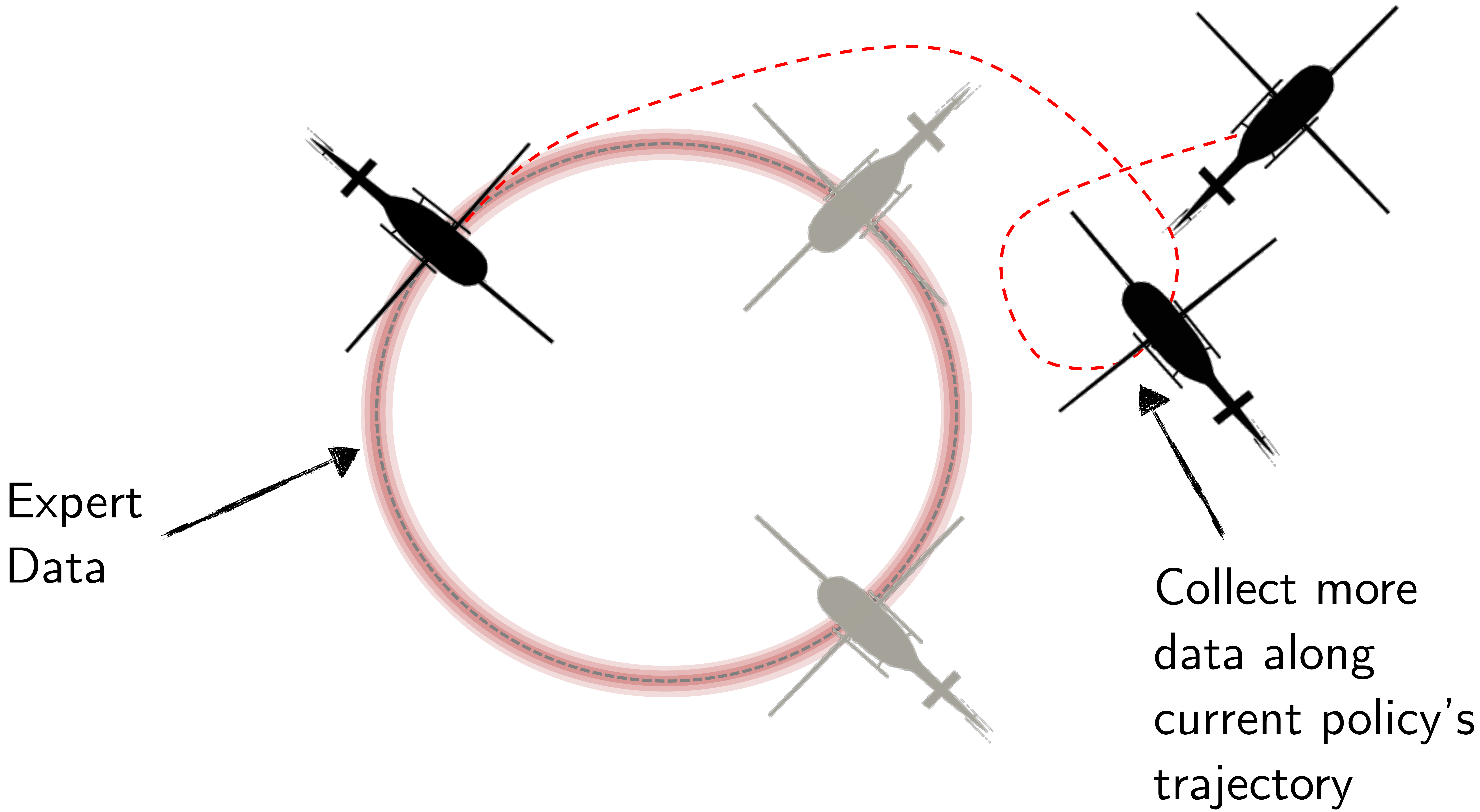
(From Ross and Bagnell, 2012)

Strategy

~~Train a model on state actions visited by the expert!~~

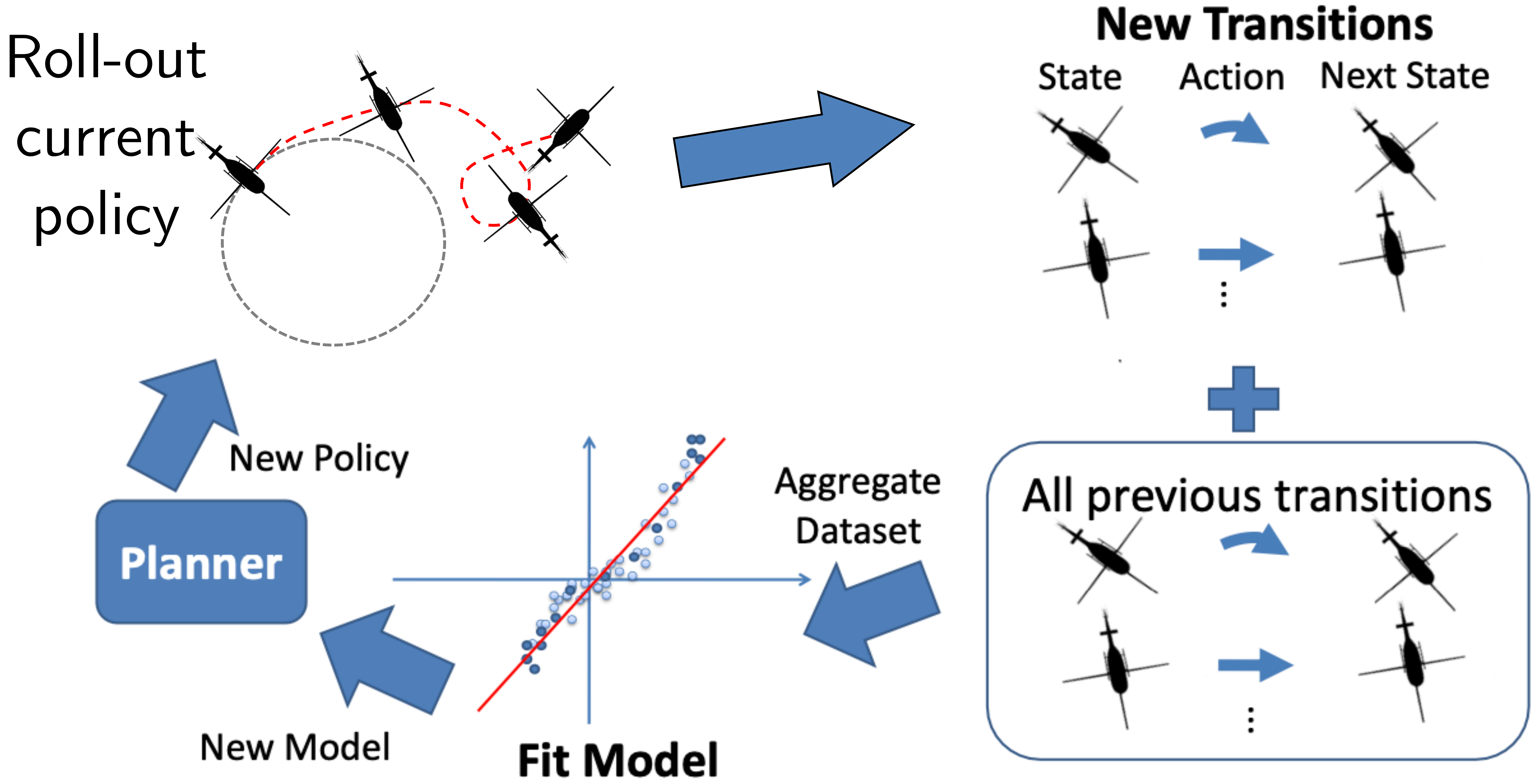
Train a model on state actions visited by the learner!

Improve model where policy goes

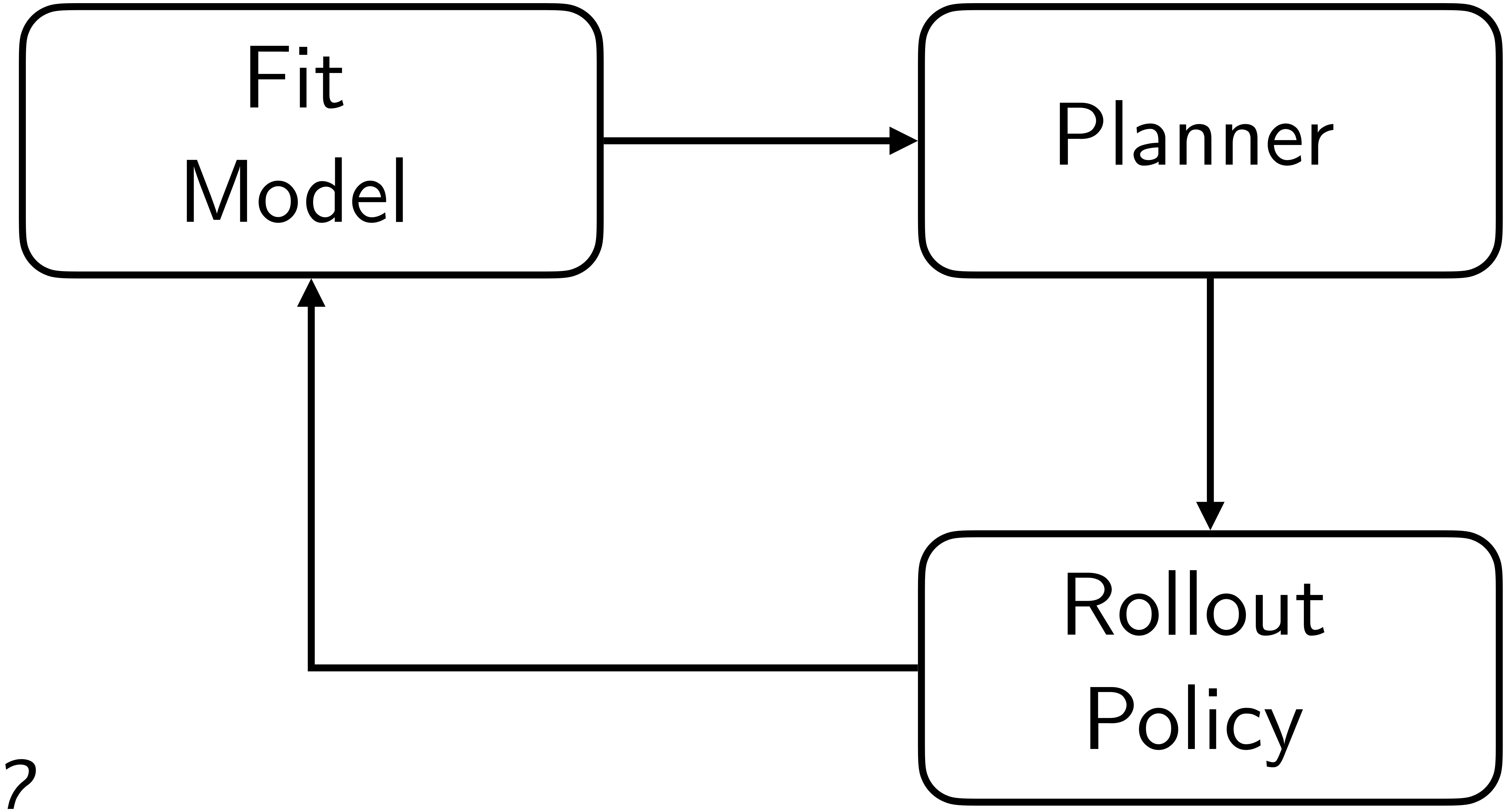


Don't we know an algorithm that does this?

DAGGER for Model-based RL!!



Model Based RL v2.0



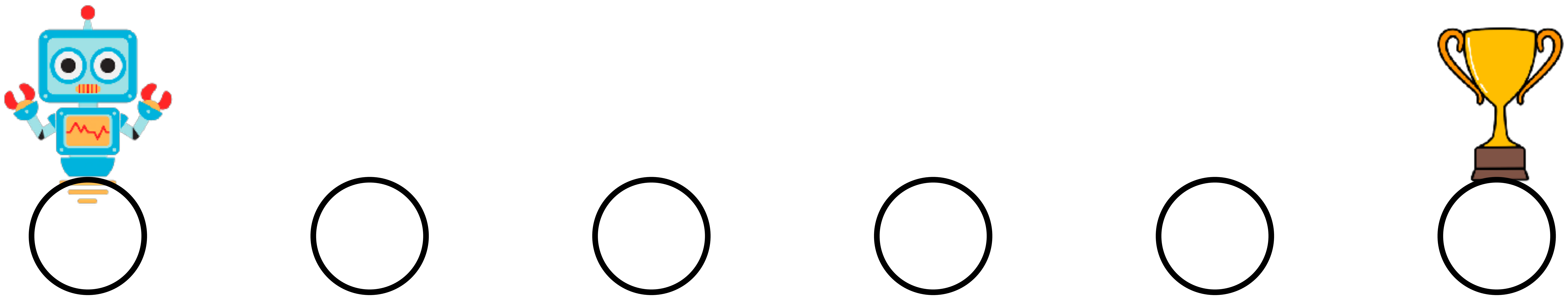
*If I **perfectly** fit a model (i.e. training error zero), this should work, right?*

Model

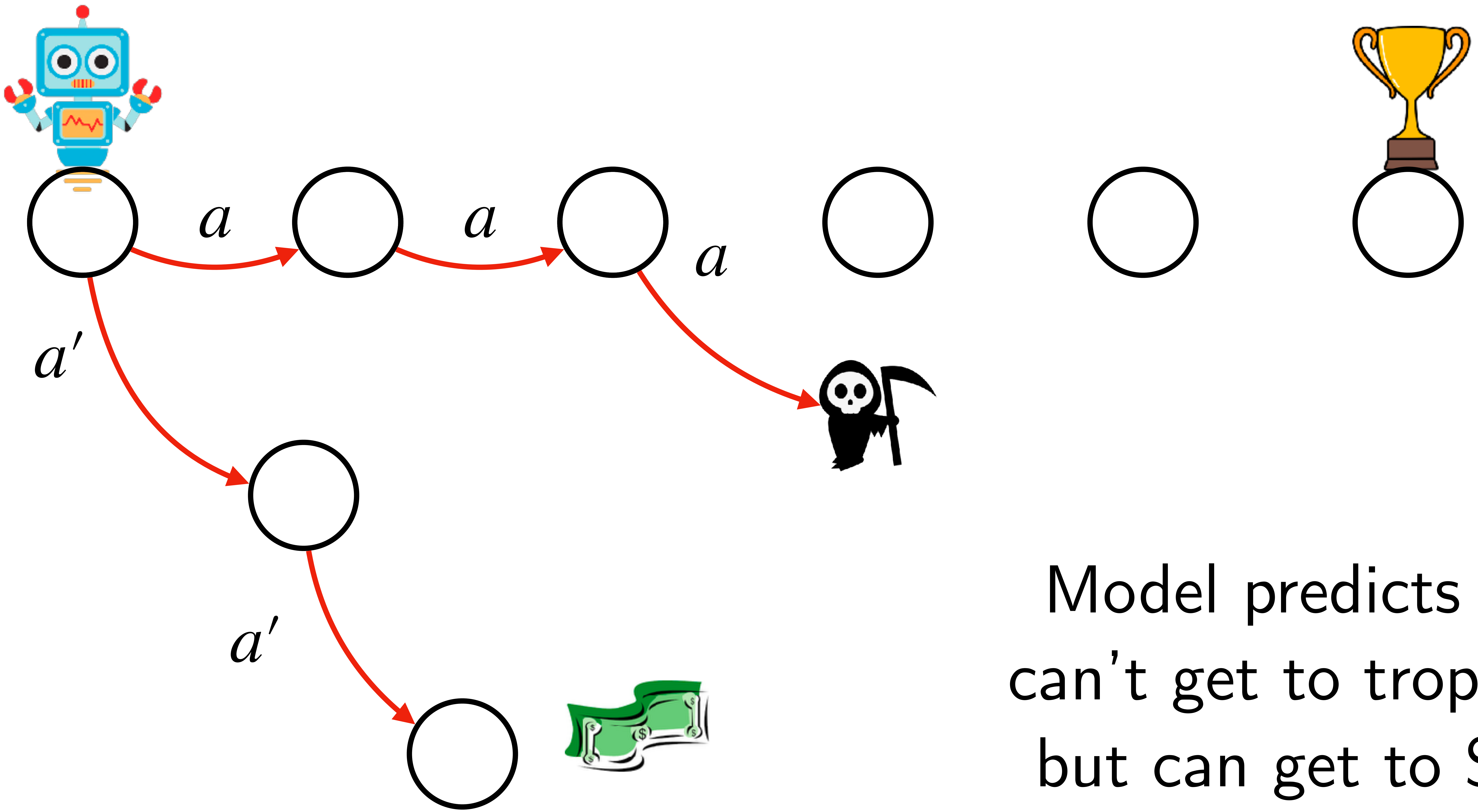
s'$=\hat{M}(s,a)$

World

s'$=M^*(s,a)$



Model $s' = \hat{M}(s, a)$ — World $s' = M^*(s, a)$



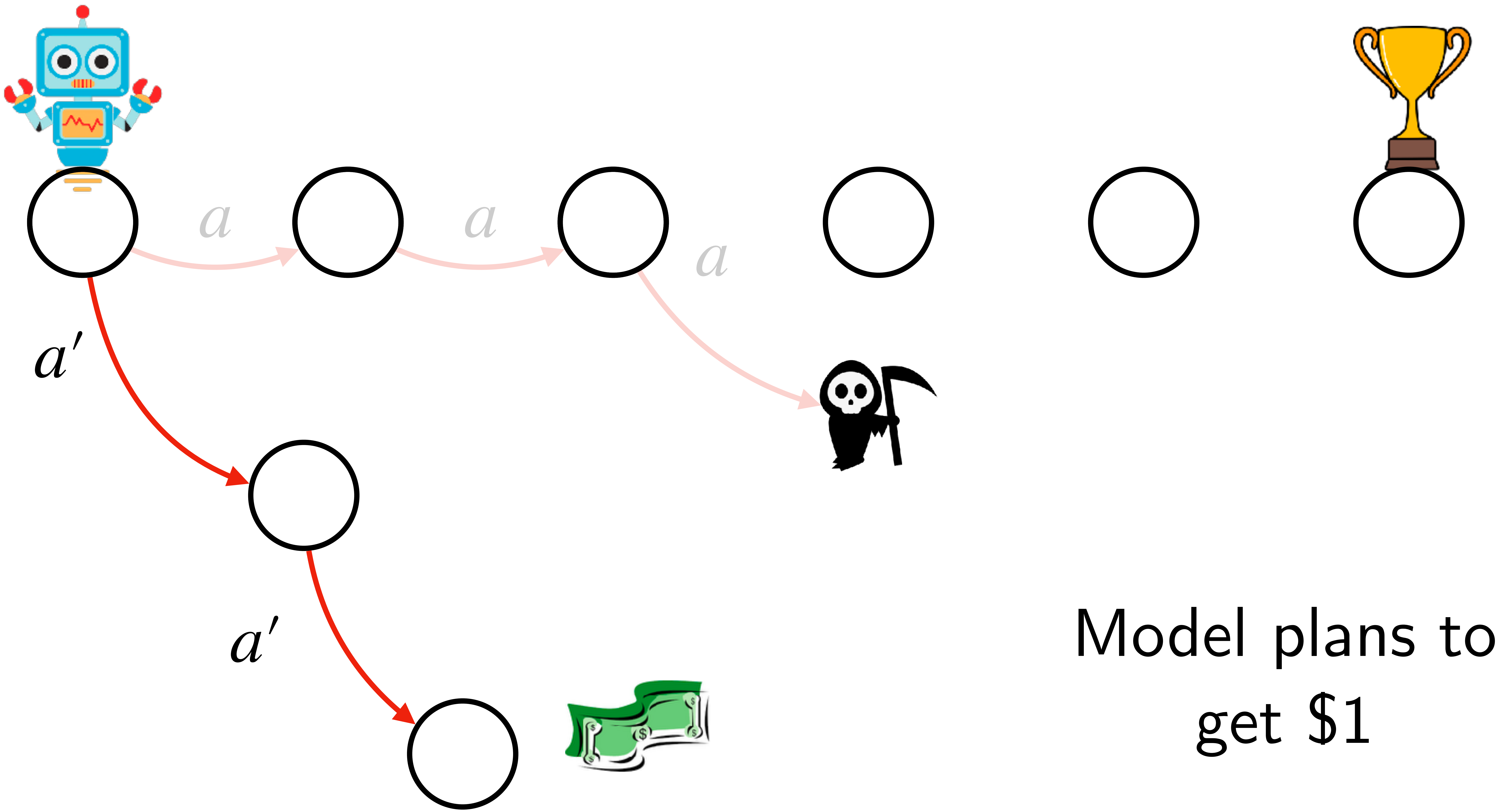
Model predicts it can't get to trophy, but can get to $1

Model

$s' = \hat{M}(s, a)$

World

$s' = M^*(s, a)$



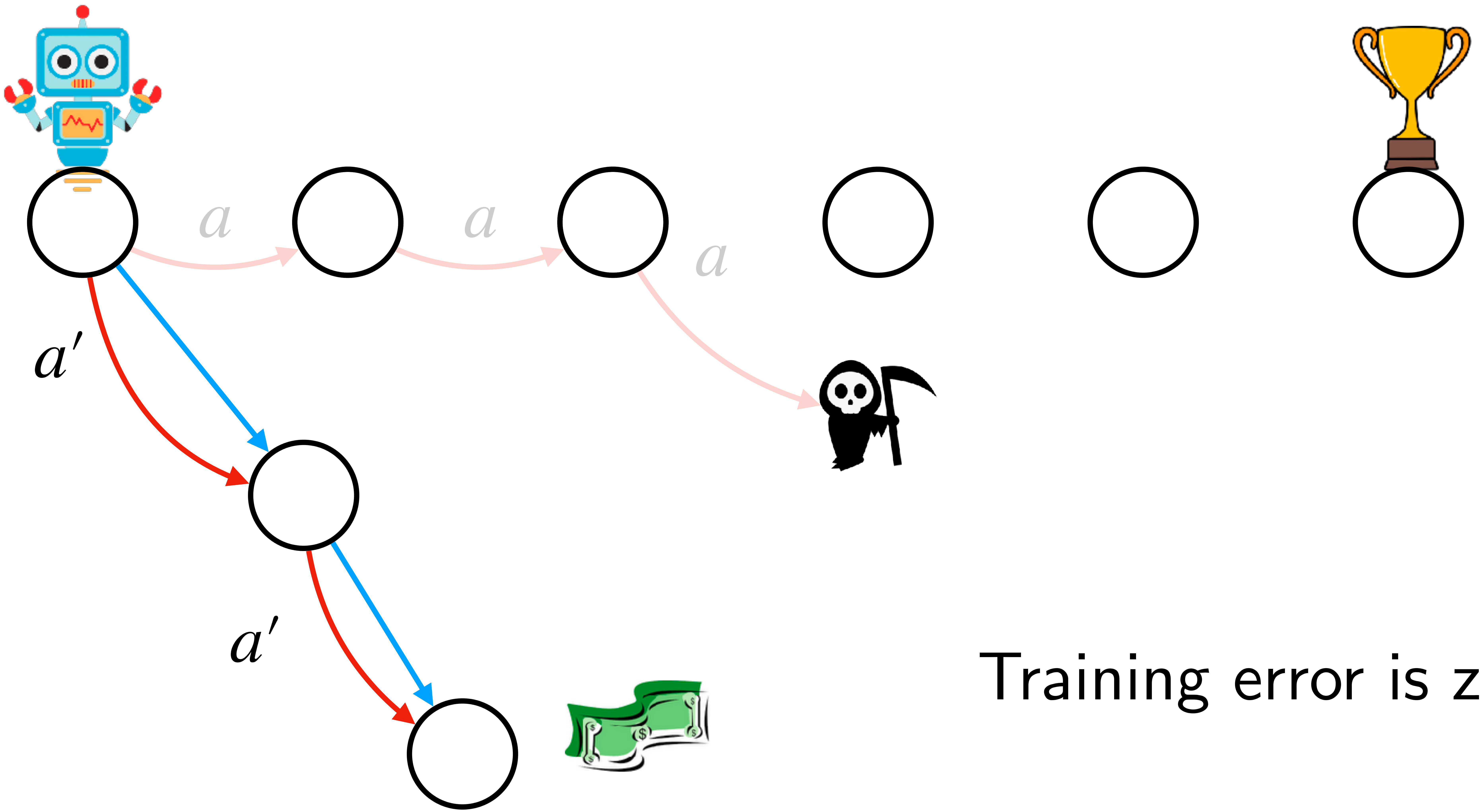
Model plans to get $1

Model

$s' = \hat{M}(s, a)$

World

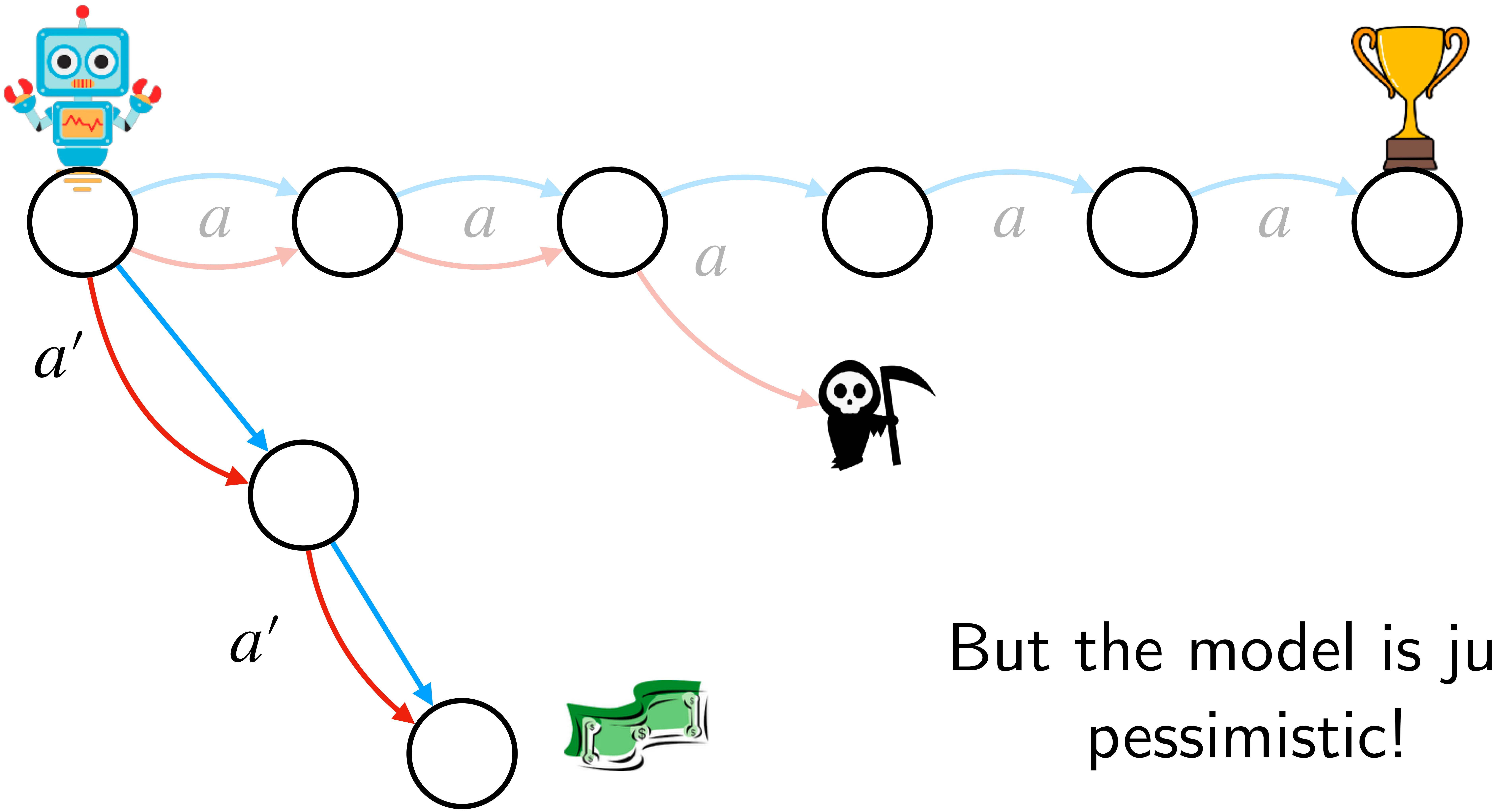
$s' = M^*(s, a)$



Training error is zero!

Model
s'=$\hat{M}(s, a)$

World
s'=$M^*(s, a)$



But the model is just pessimistic!

Strategy

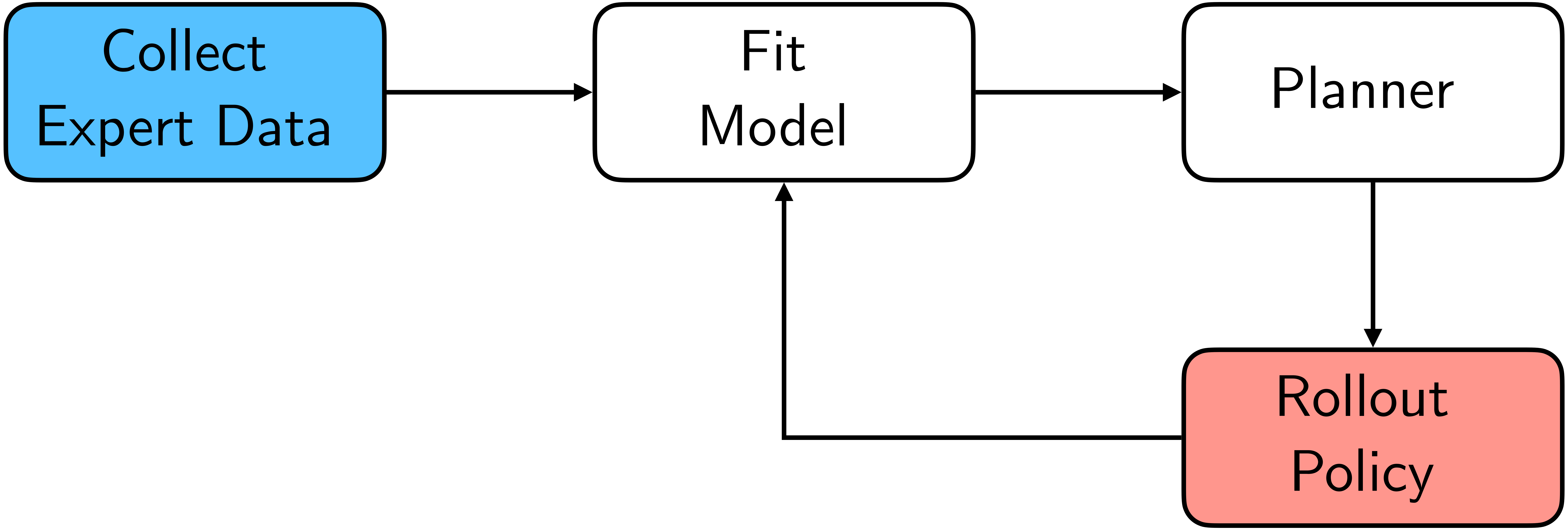
~~Train a model on state actions visited by the expert!~~

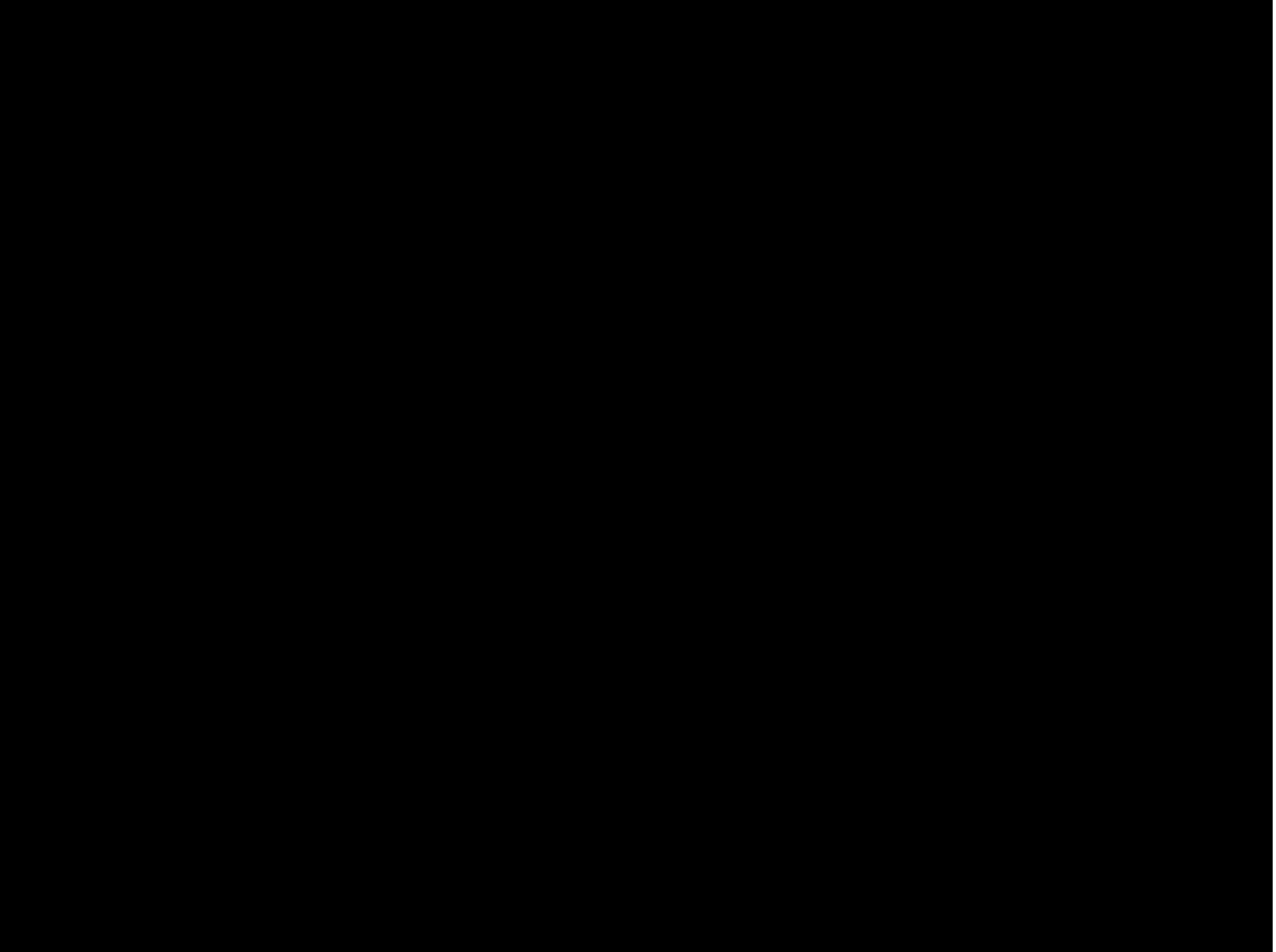
~~Train a model on state actions visited by the learner!~~

Train a model on state actions visited by *both* the expert and the learner!

Model Learning with Planner in Loop

(Ross & Bagnell, 2012)





Model learning on both expert and learner data works!

(From Ross & Bagnell, 2012)

How do we derive this strategy?

Theoretical Foundations for Model Based RL

Agnostic System Identification for Model-Based Reinforcement Learning

Stéphane Ross STEPHANEROSS@CMU.EDU
Robotics Institute, Carnegie Mellon University, PA USA

J. Andrew Bagnell DBAGNELL@RI.CMU.EDU
Robotics Institute, Carnegie Mellon University, PA USA

Lemma: Performance Difference via Planning in Model

$$J_{M^*}(\pi^*) - J_{M^*}(\hat{\pi})$$

$$\leq \mathbb{E}_{s_0}\left[V^{\hat{\pi}}_{\hat{M}}(s_0) - V^{\pi^*}_{\hat{M}}(s_0)\right] \quad + TV_{\max}\mathbb{E}_{s,a\sim\pi^*}||\hat{M}(s,a) - M^*(s,a)||$$

Planning error *Model fit on expert states*

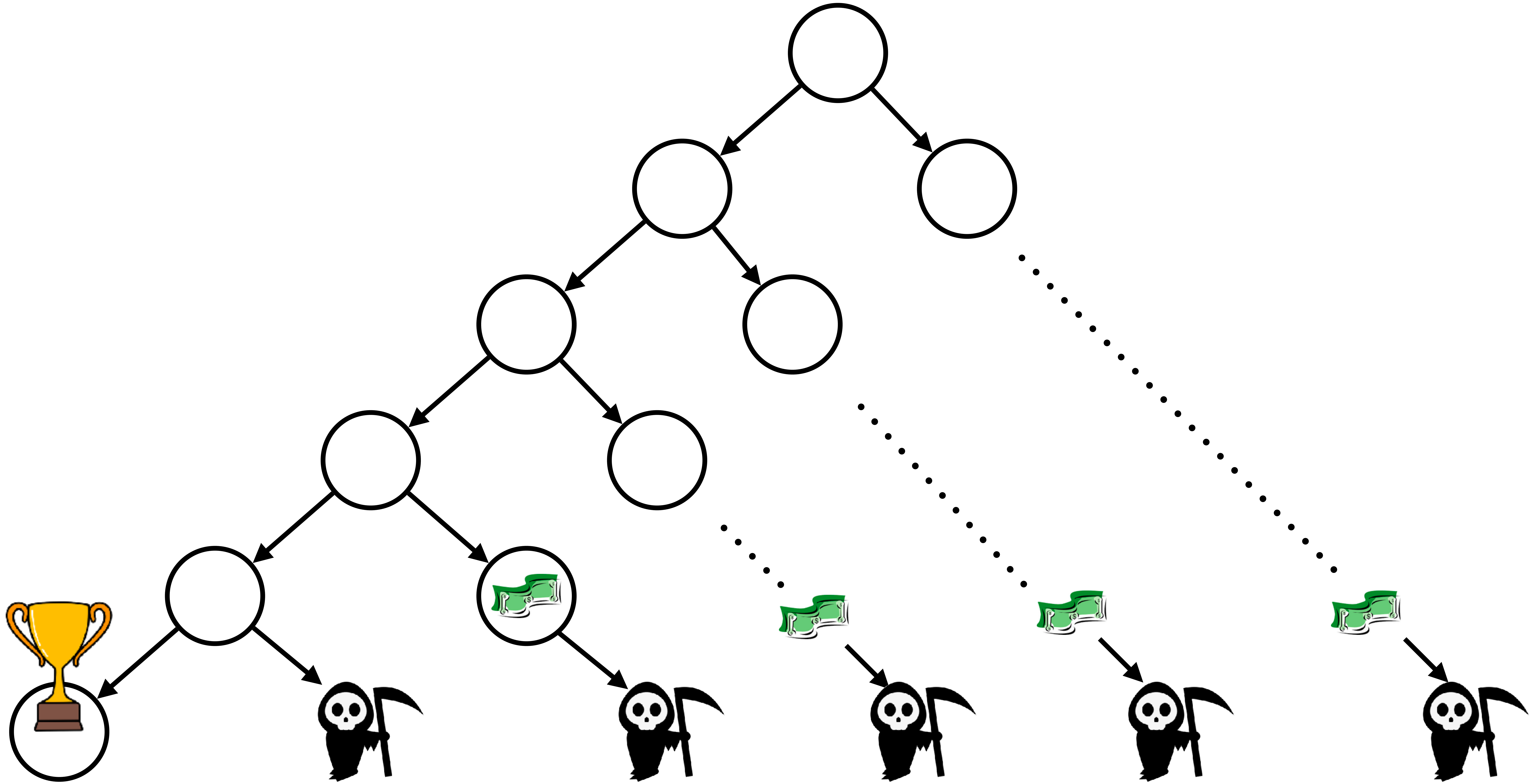
$$+ TV_{\max}\mathbb{E}_{s,a\sim\hat{\pi}}||\hat{M}(s,a) - M^*(s,a)||$$

Model fit on policy states

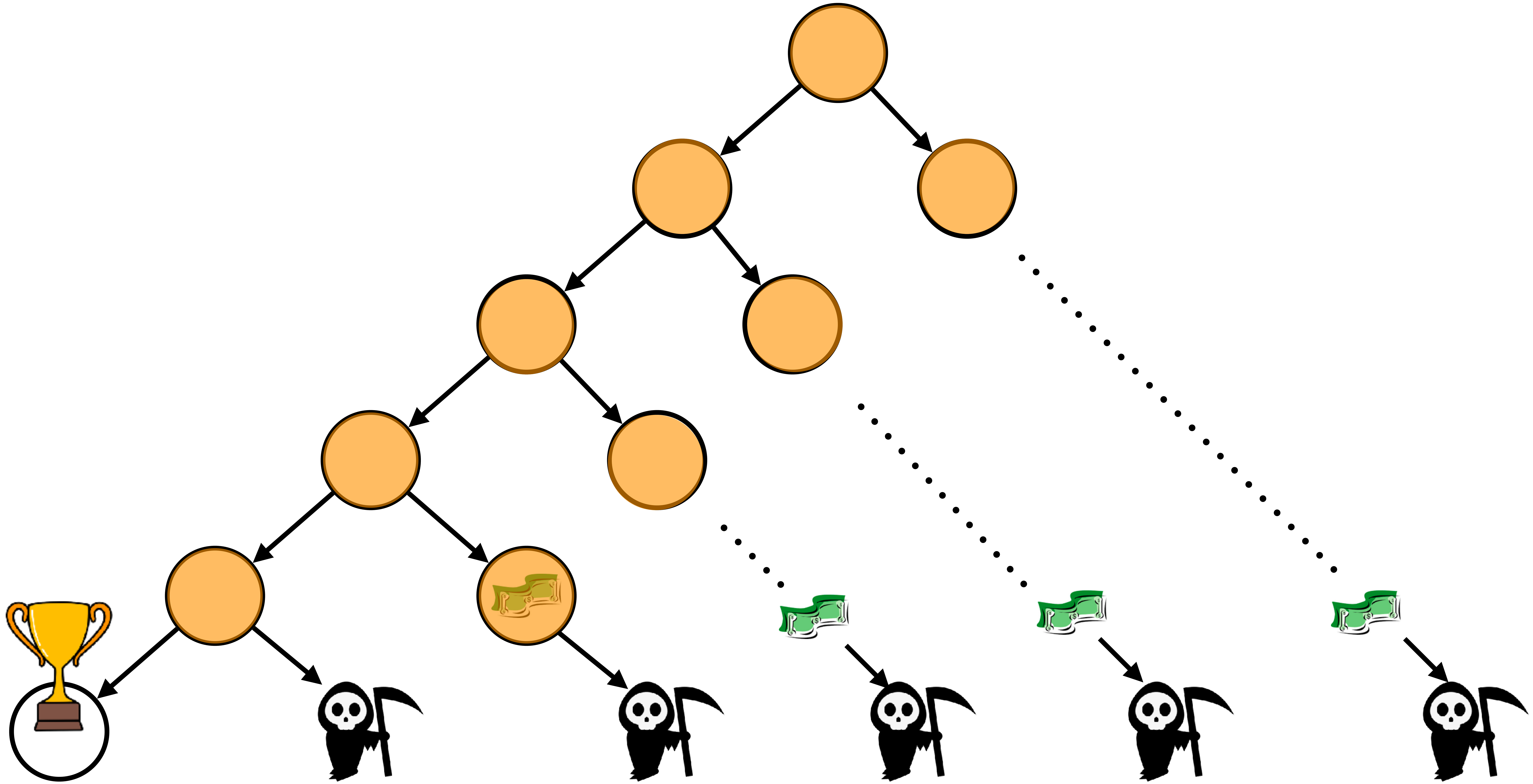
The Challenge.


Planning is like finding a
needle in an exponential haystack

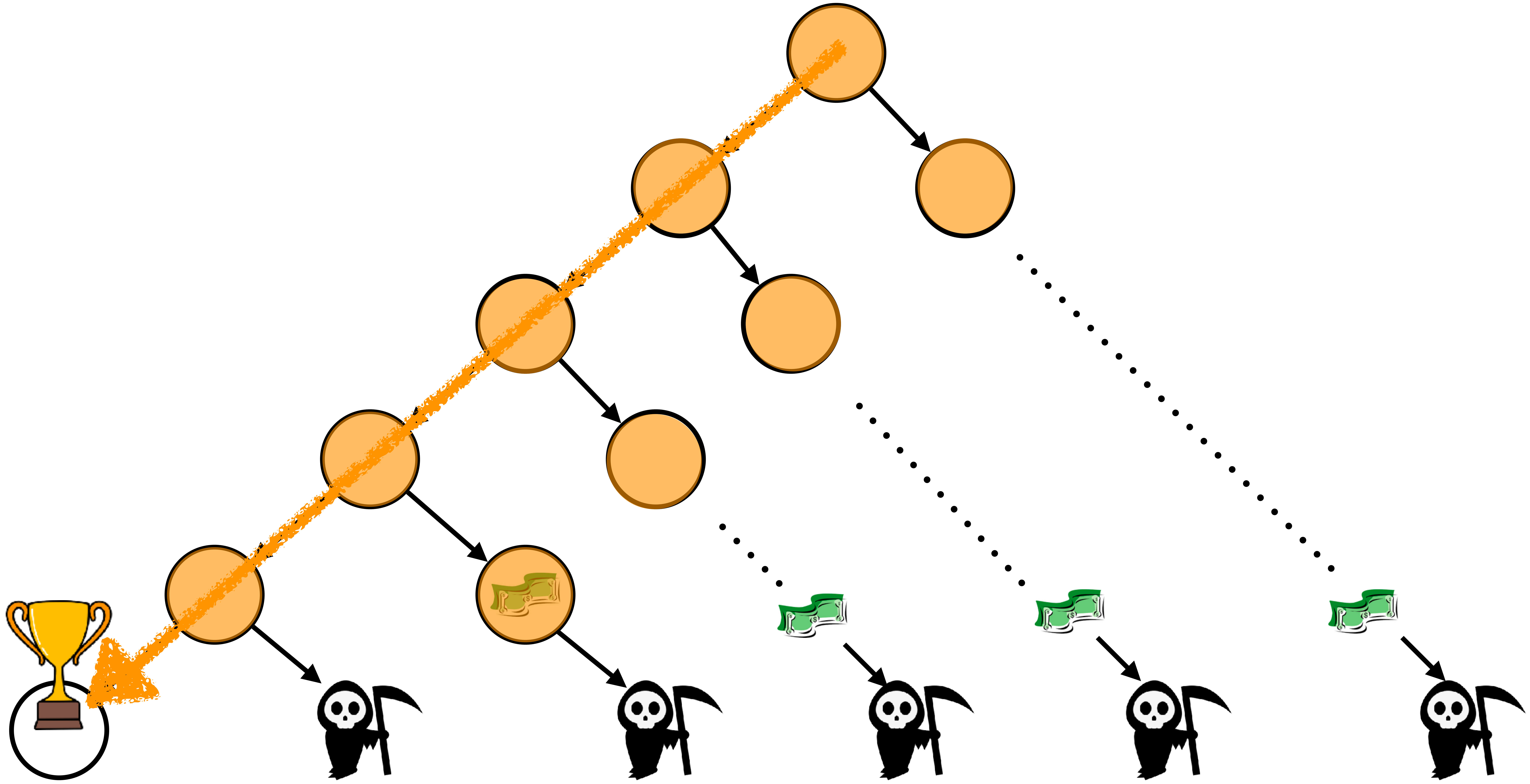
A Tree MDP



Planning is exp(T)!

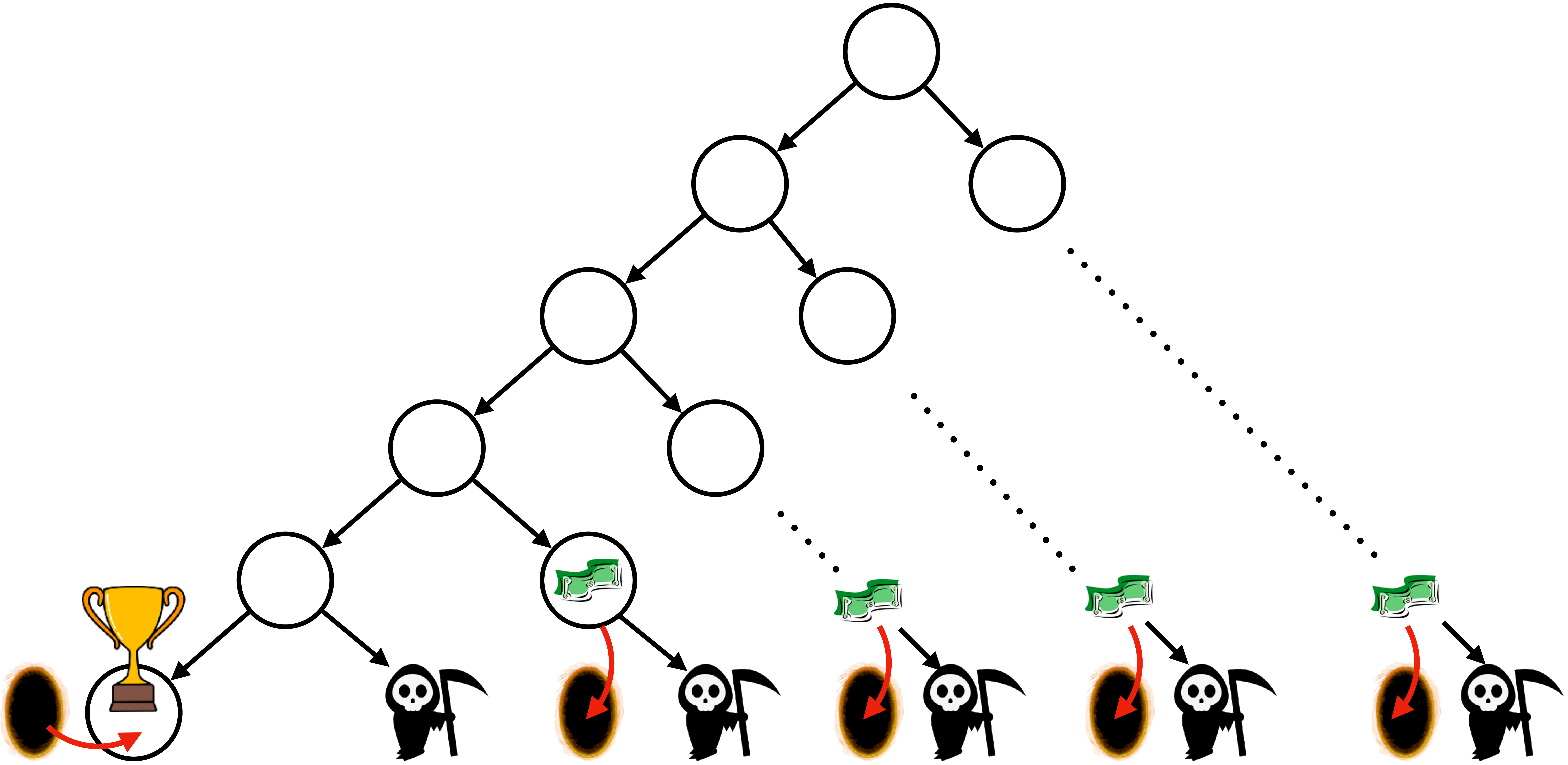


Planning is exp(T)!

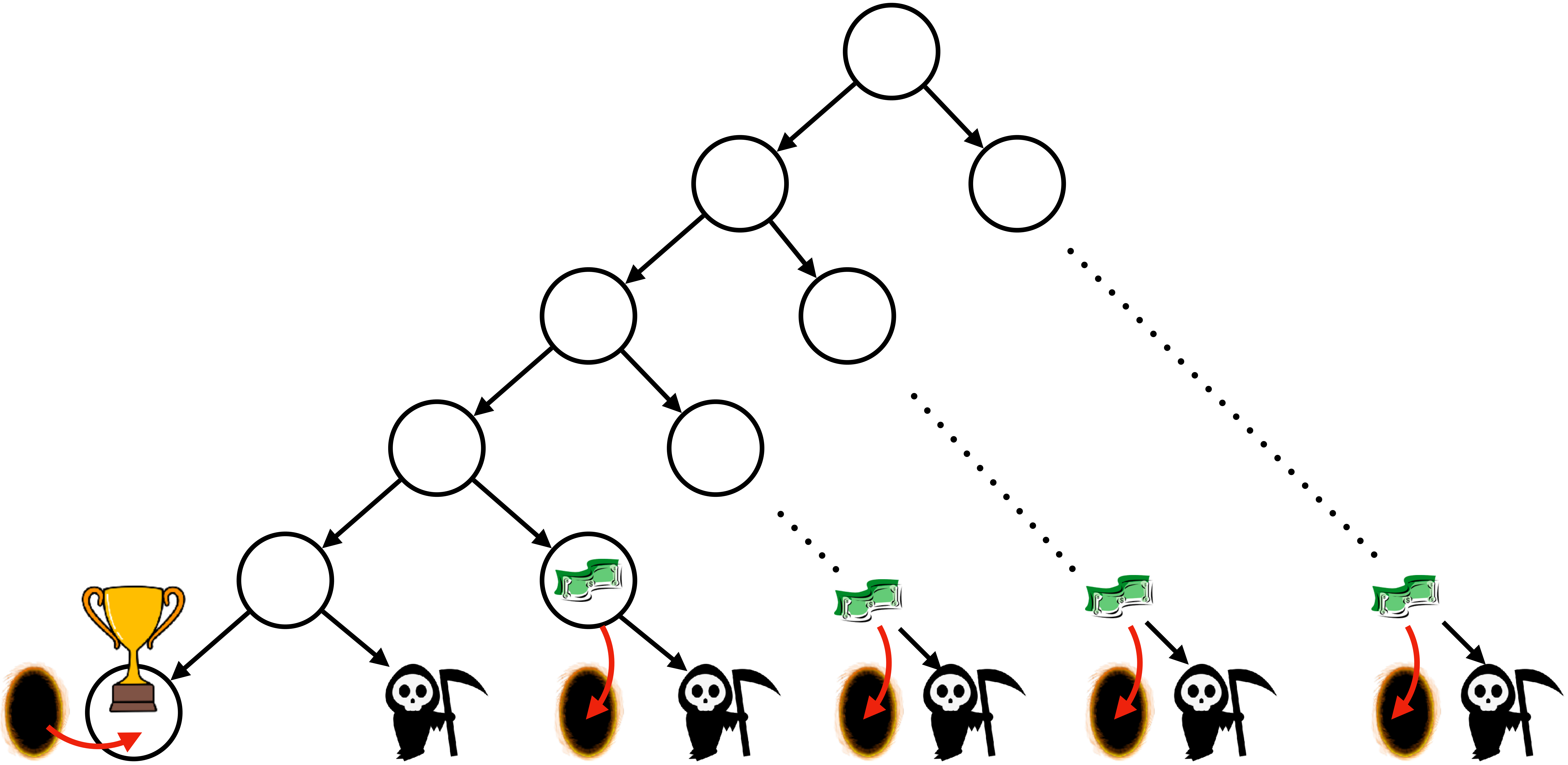


How much planning do we need when learning models?

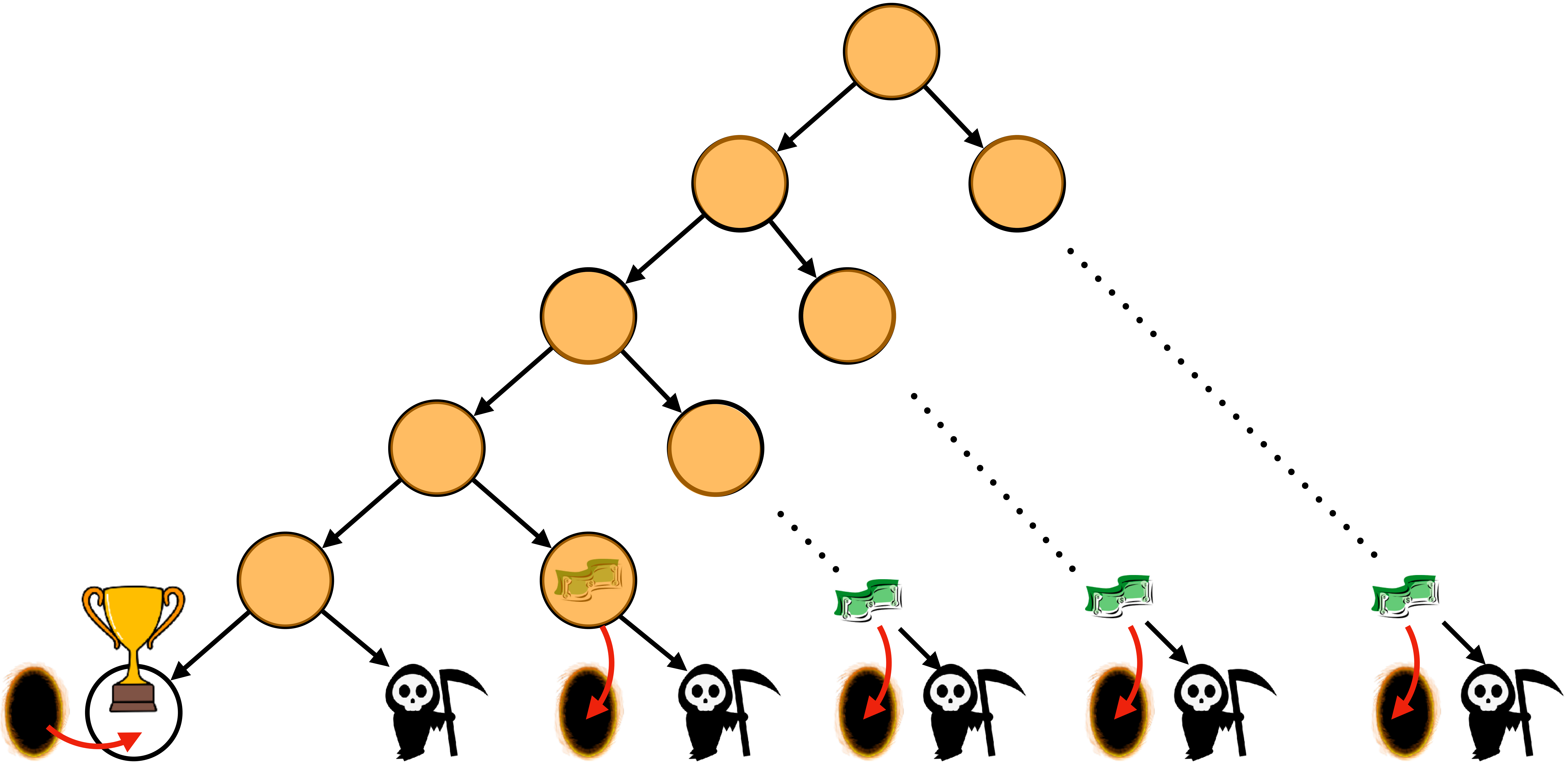
Learnt model has hidden portals!



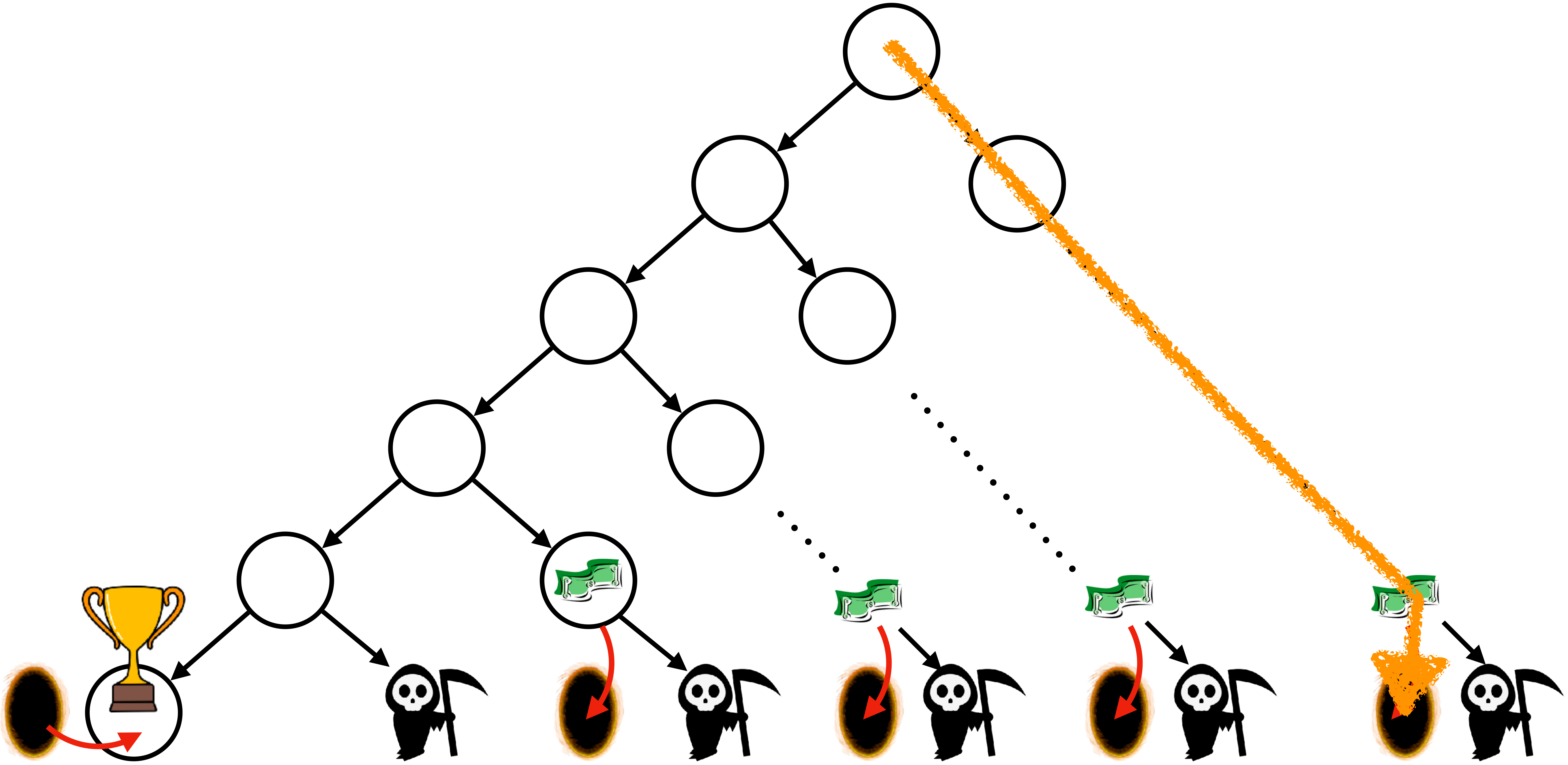
Model at iteration 0



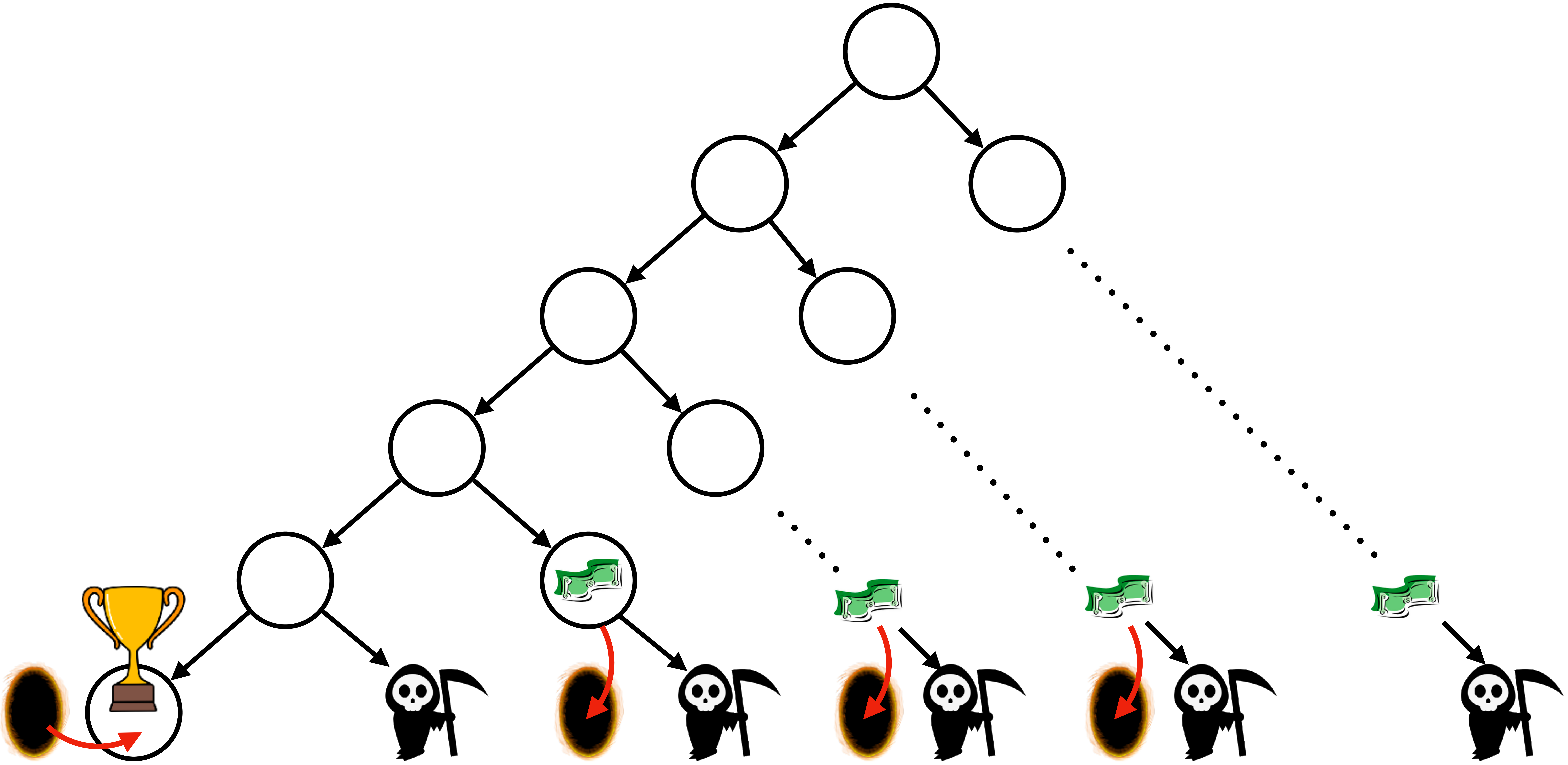
Run planning for exp(T)



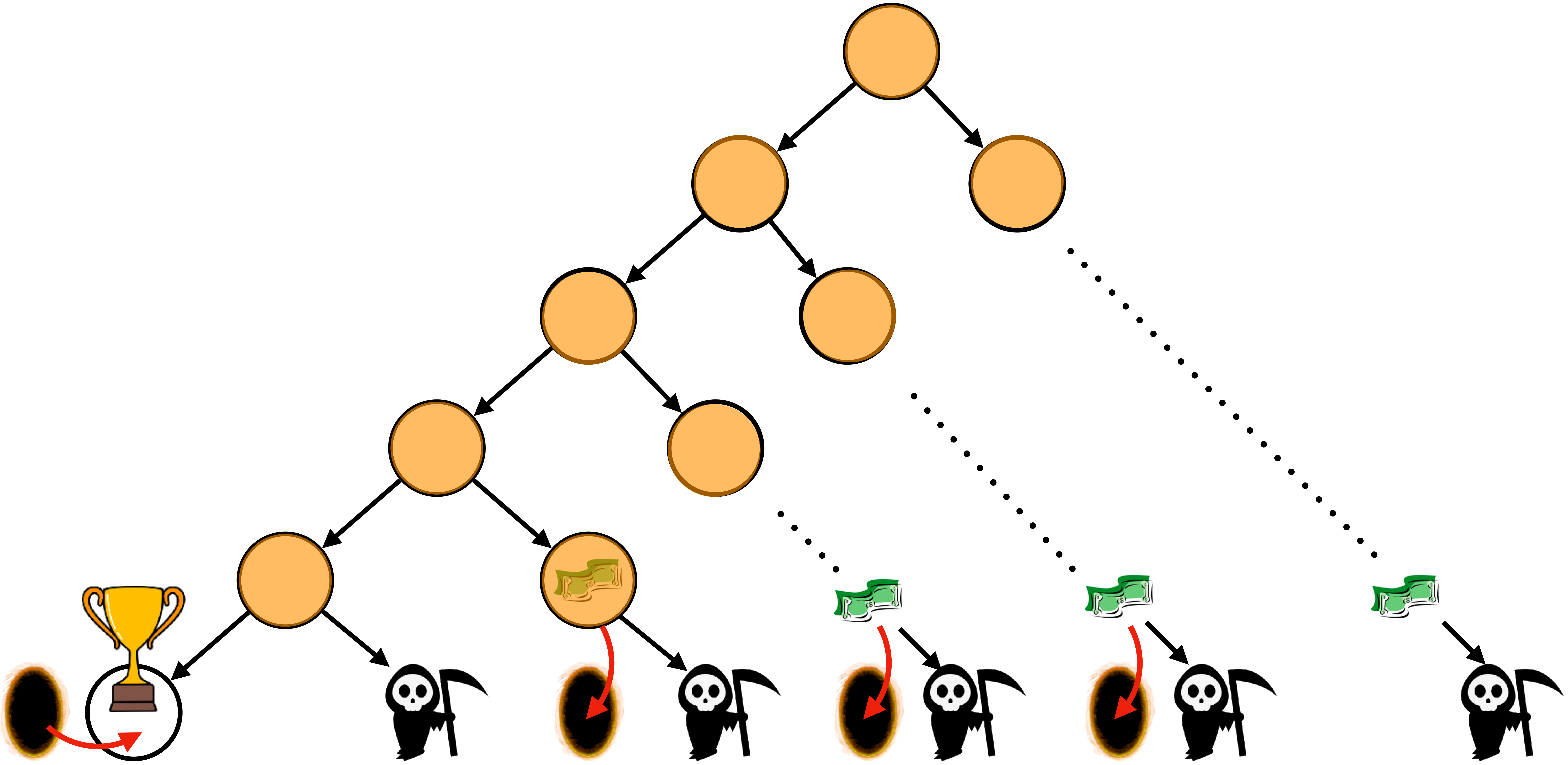
Policy at iteration 0



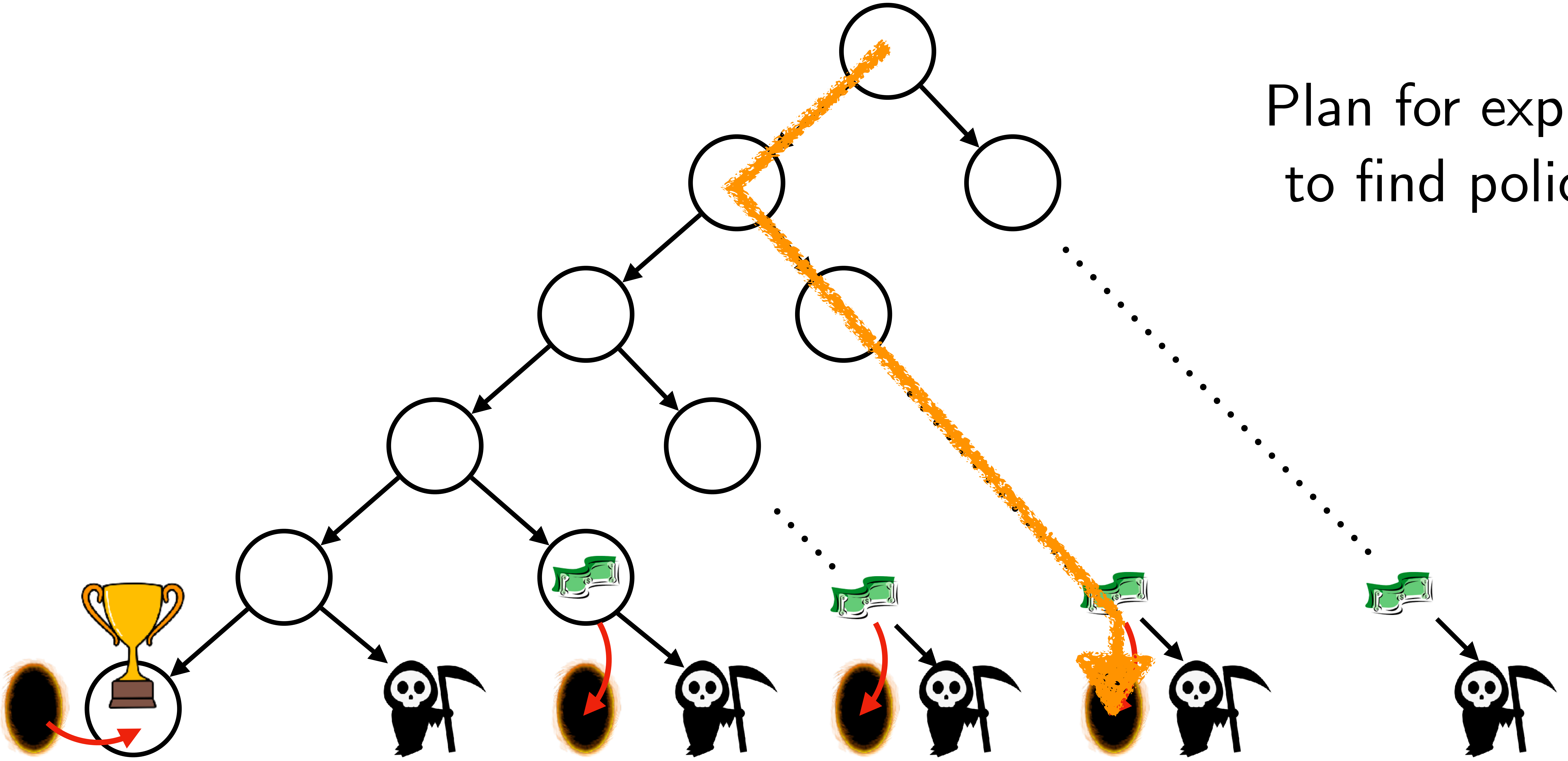
Model at iteration 1



Run planning for exp(T)

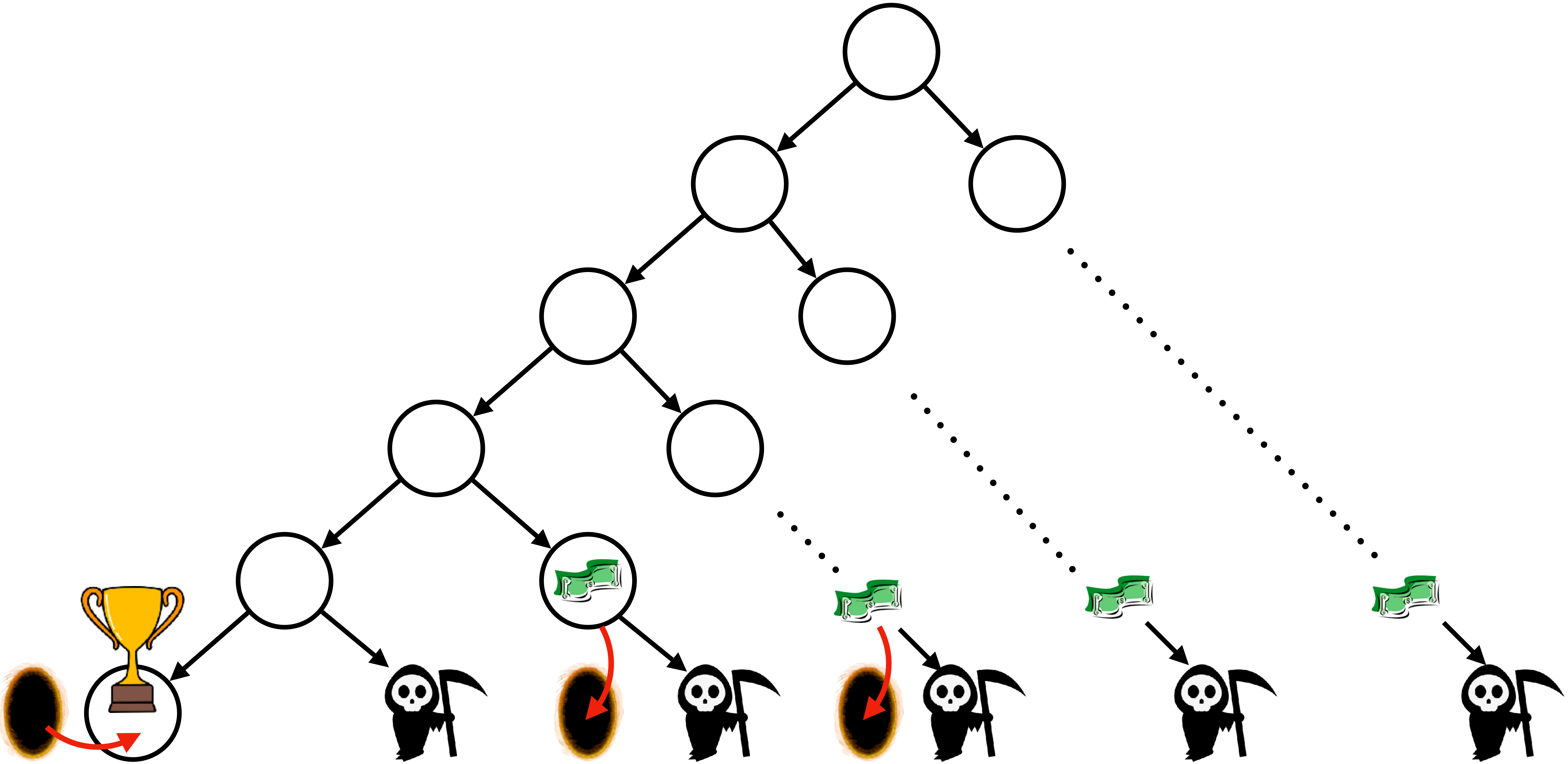


Policy at iteration 1

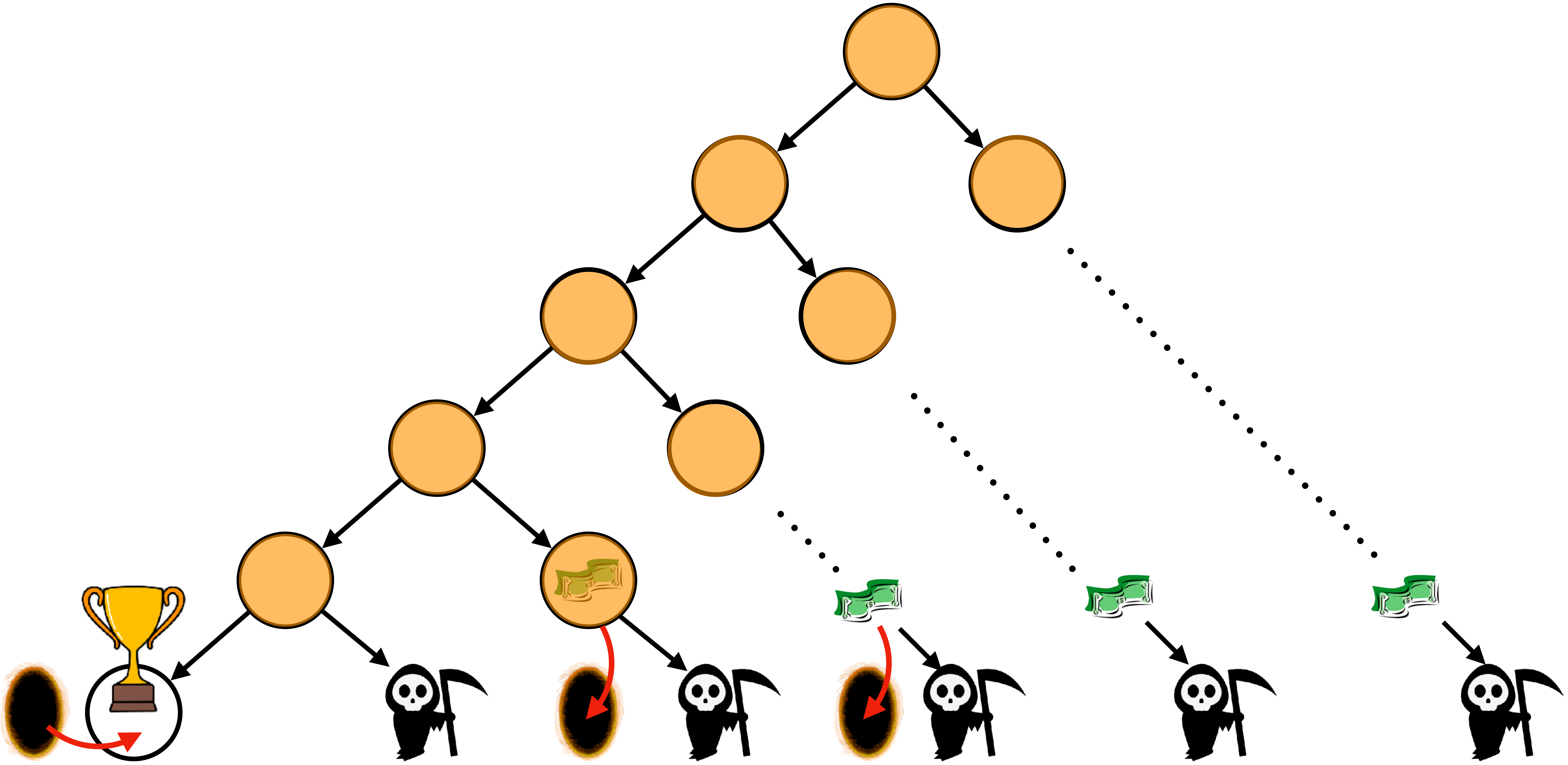


Plan for exp(T)
to find policy!

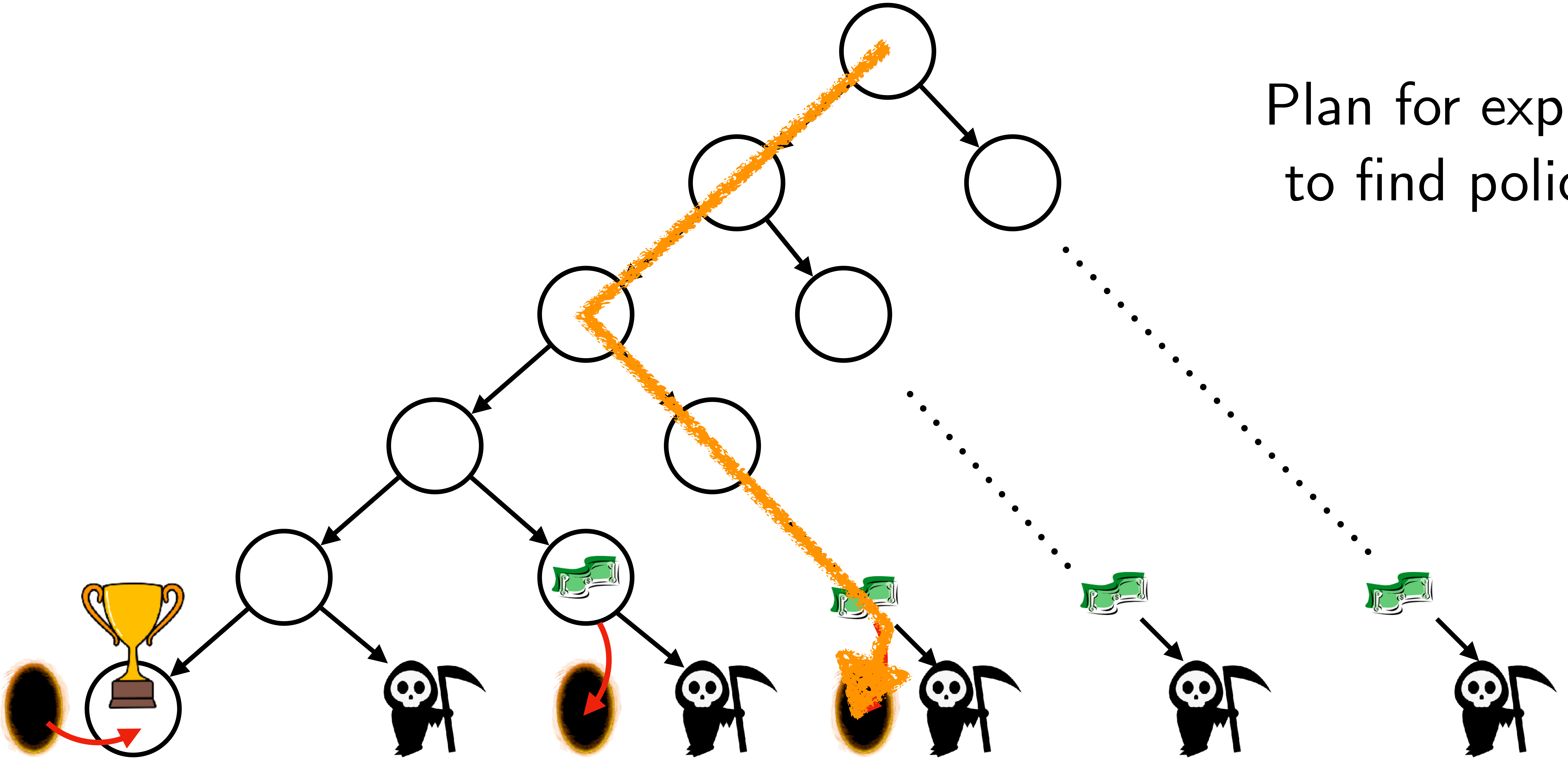
Model at iteration 2



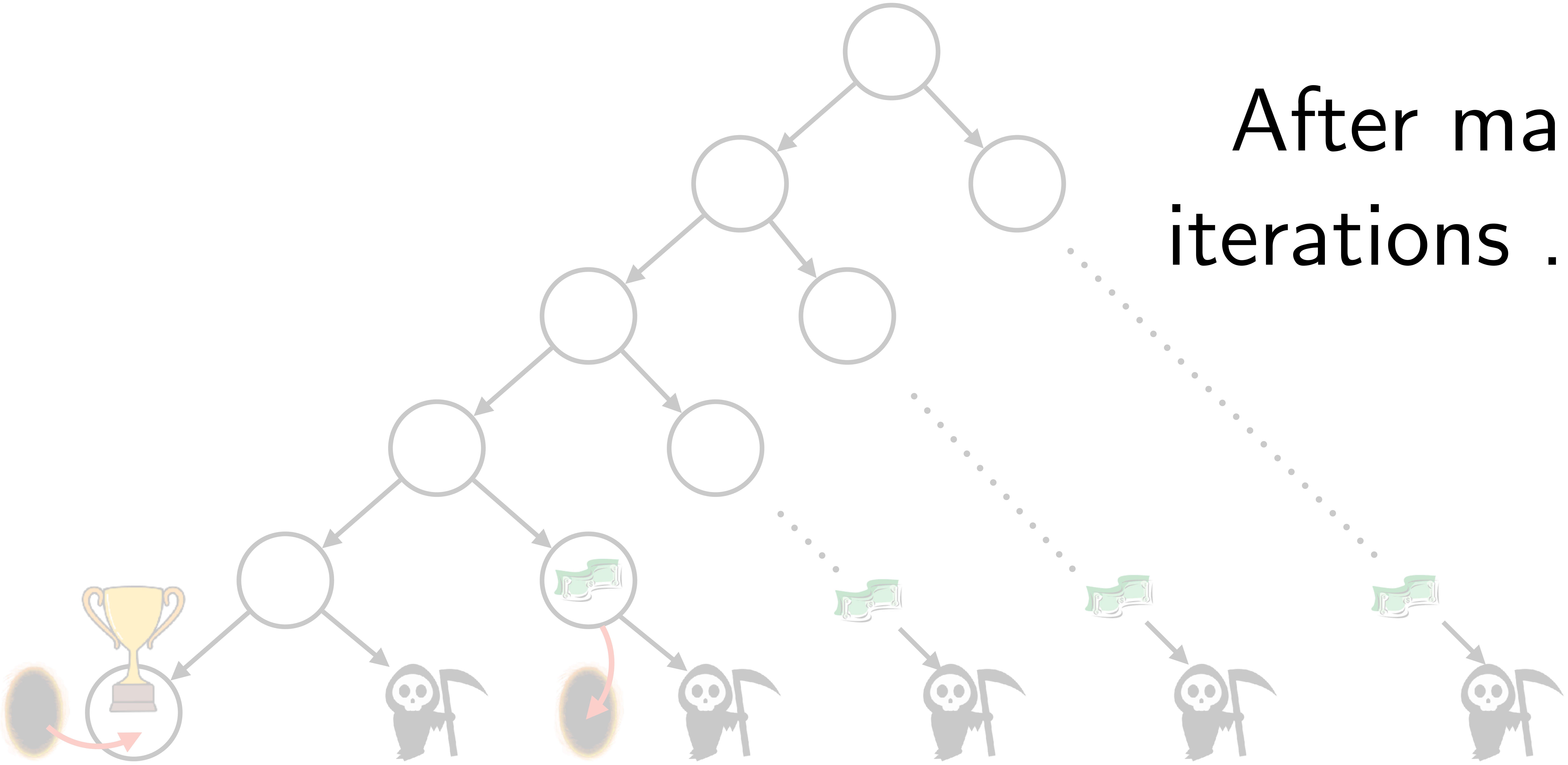
Run planning for exp(T)



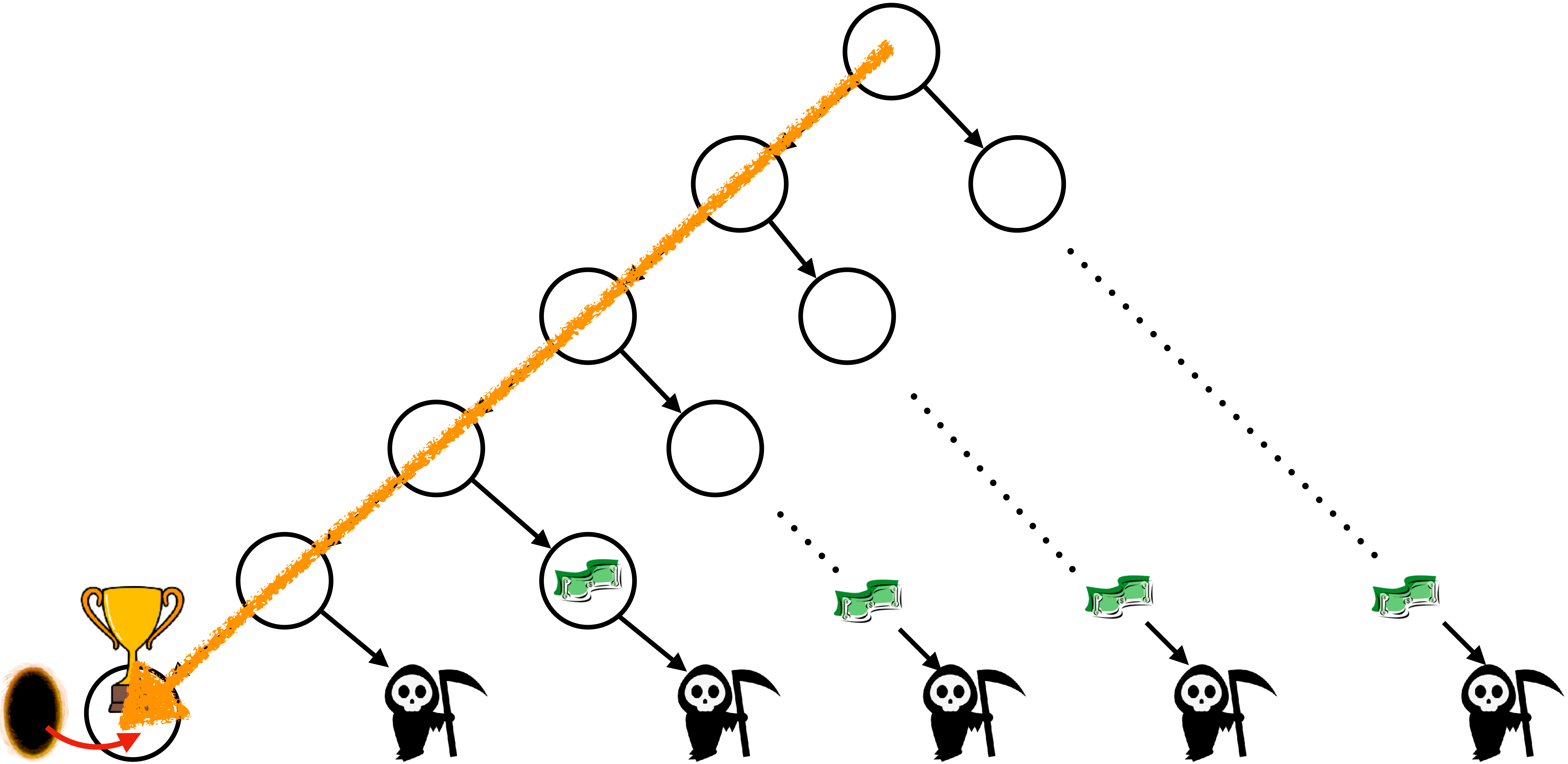
Policy at iteration 2



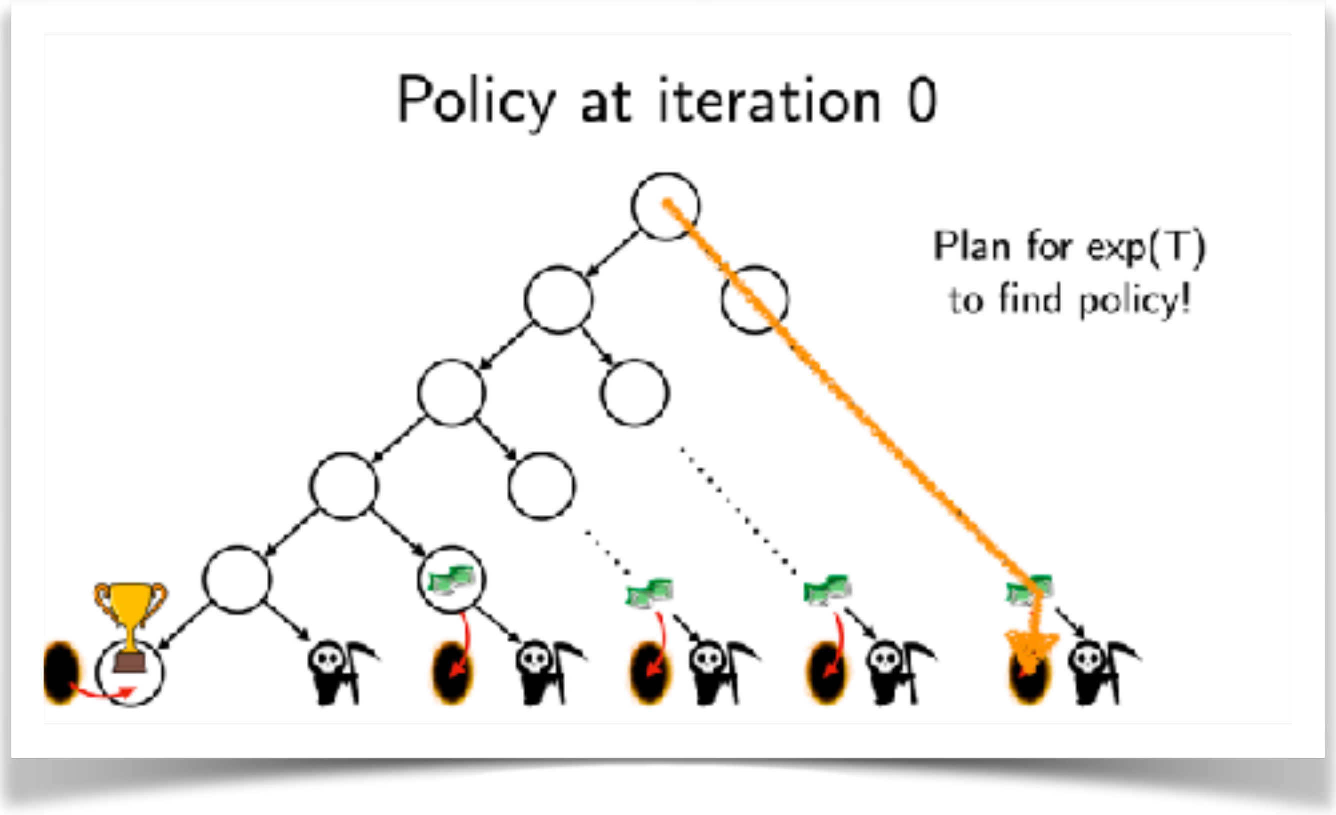
Plan for exp(T)
to find policy!

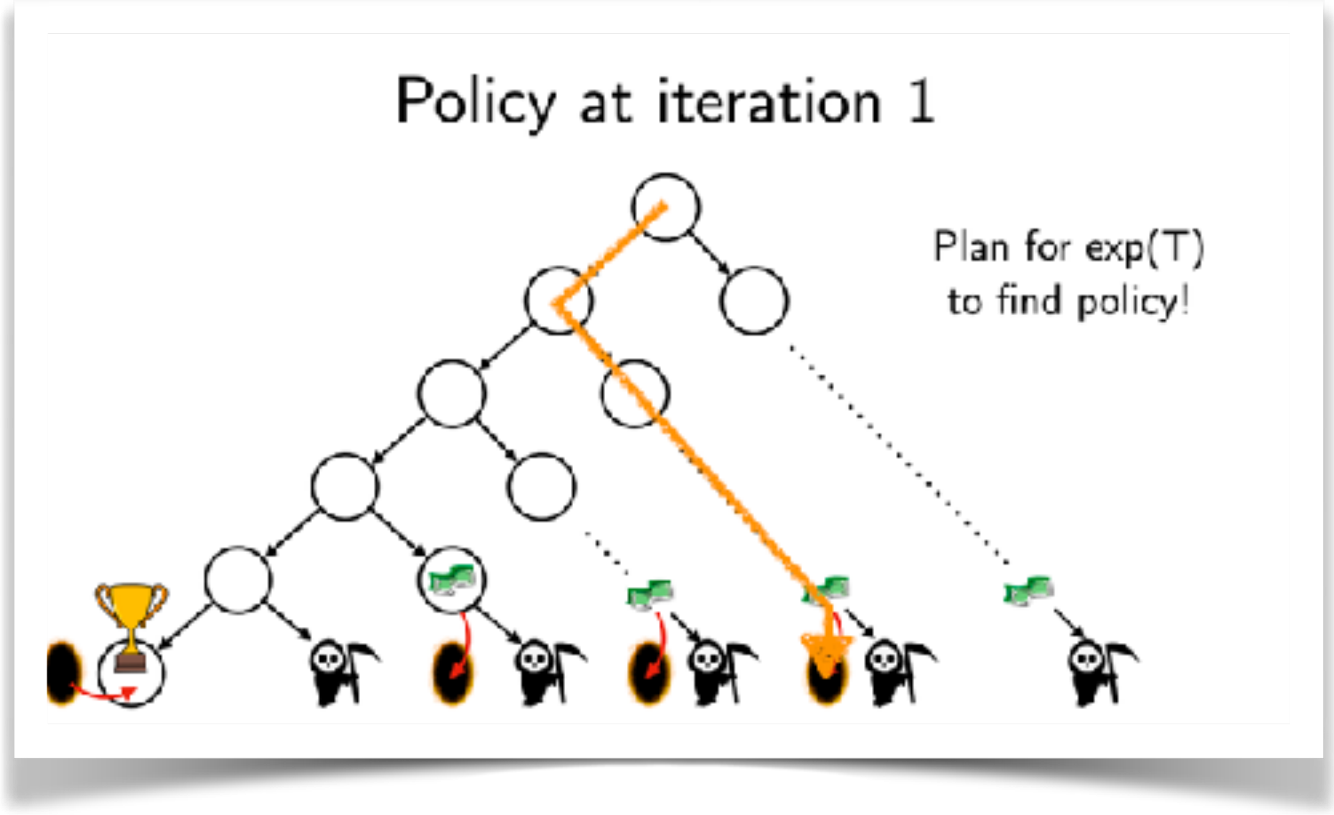


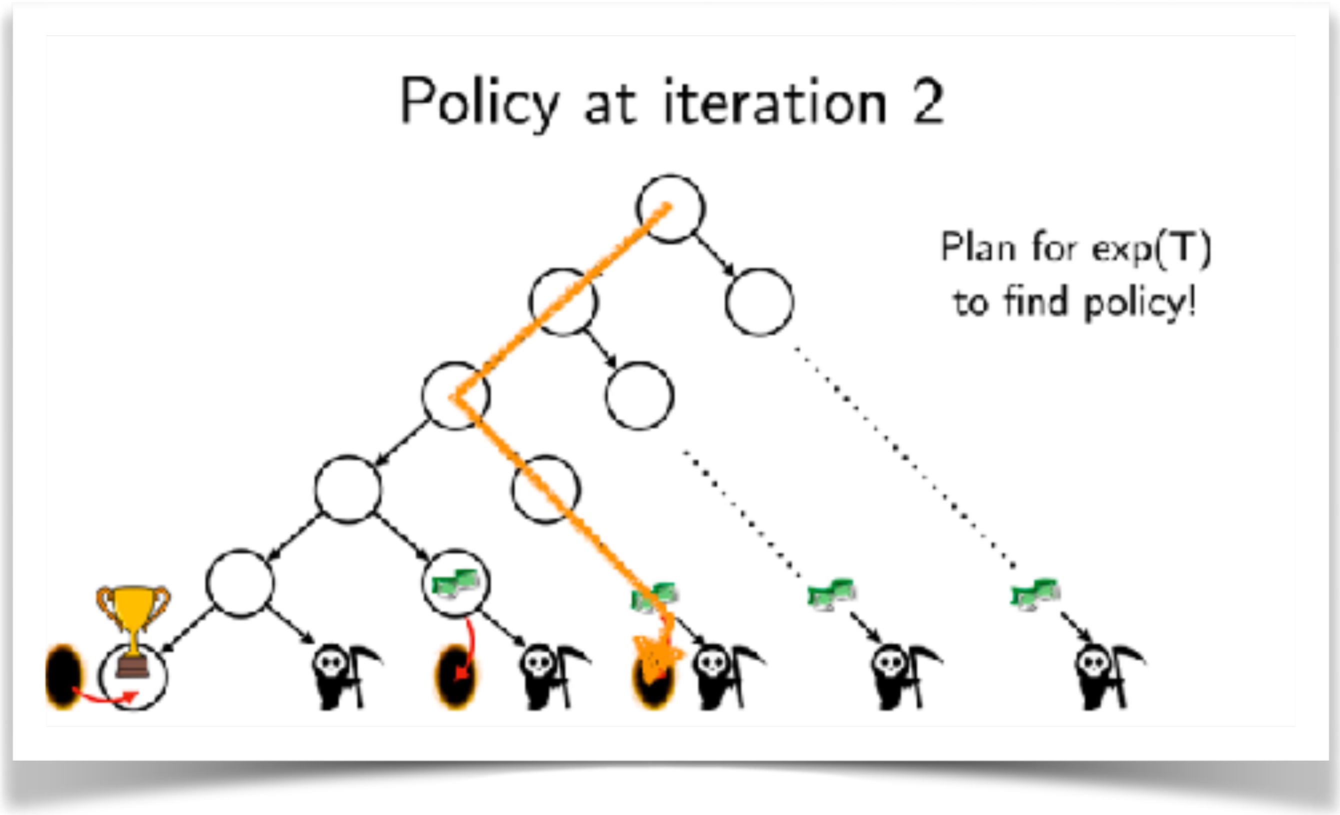
After many iterations ....



Exponential Complexity of Model Learning







.....

Every iteration, planning is exp(T) computation

Repeat for many iterations to eliminate all portals

Key Insight.

Be Lazy.

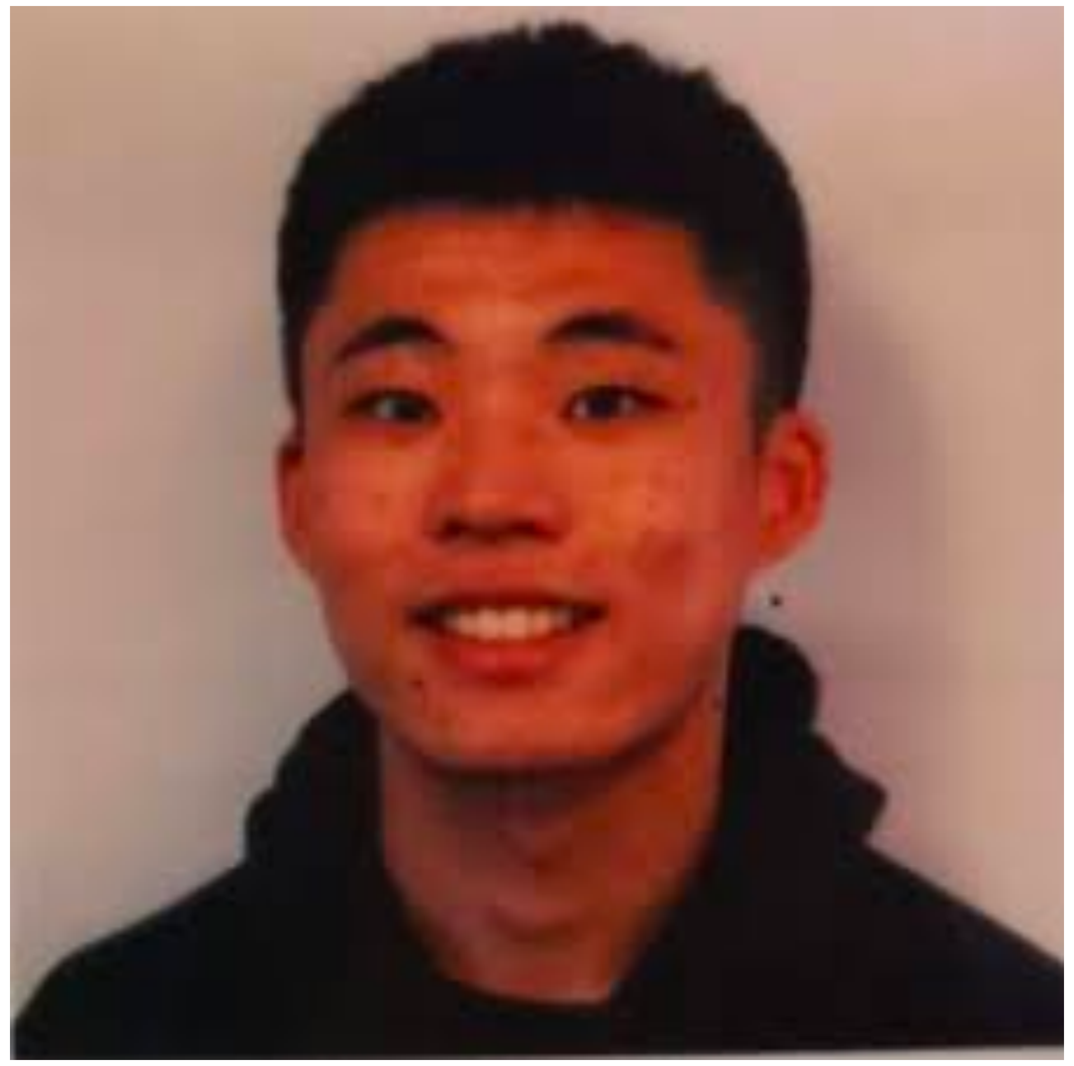
Don't compute optimal plan.

Just do better than expert.

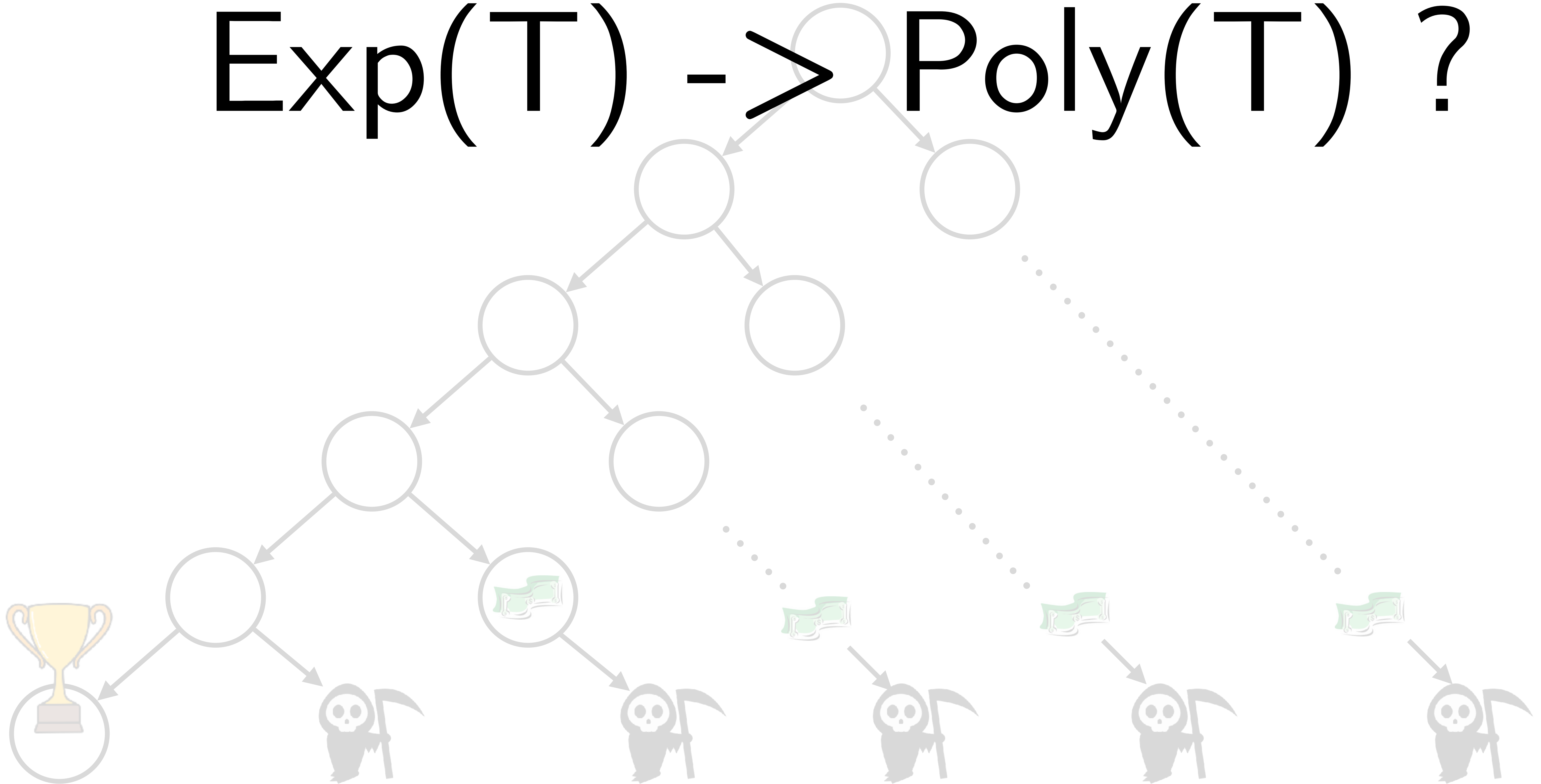
ICML 2023!

The Virtues of Laziness in Model-based RL: A Unified Objective and Algorithms

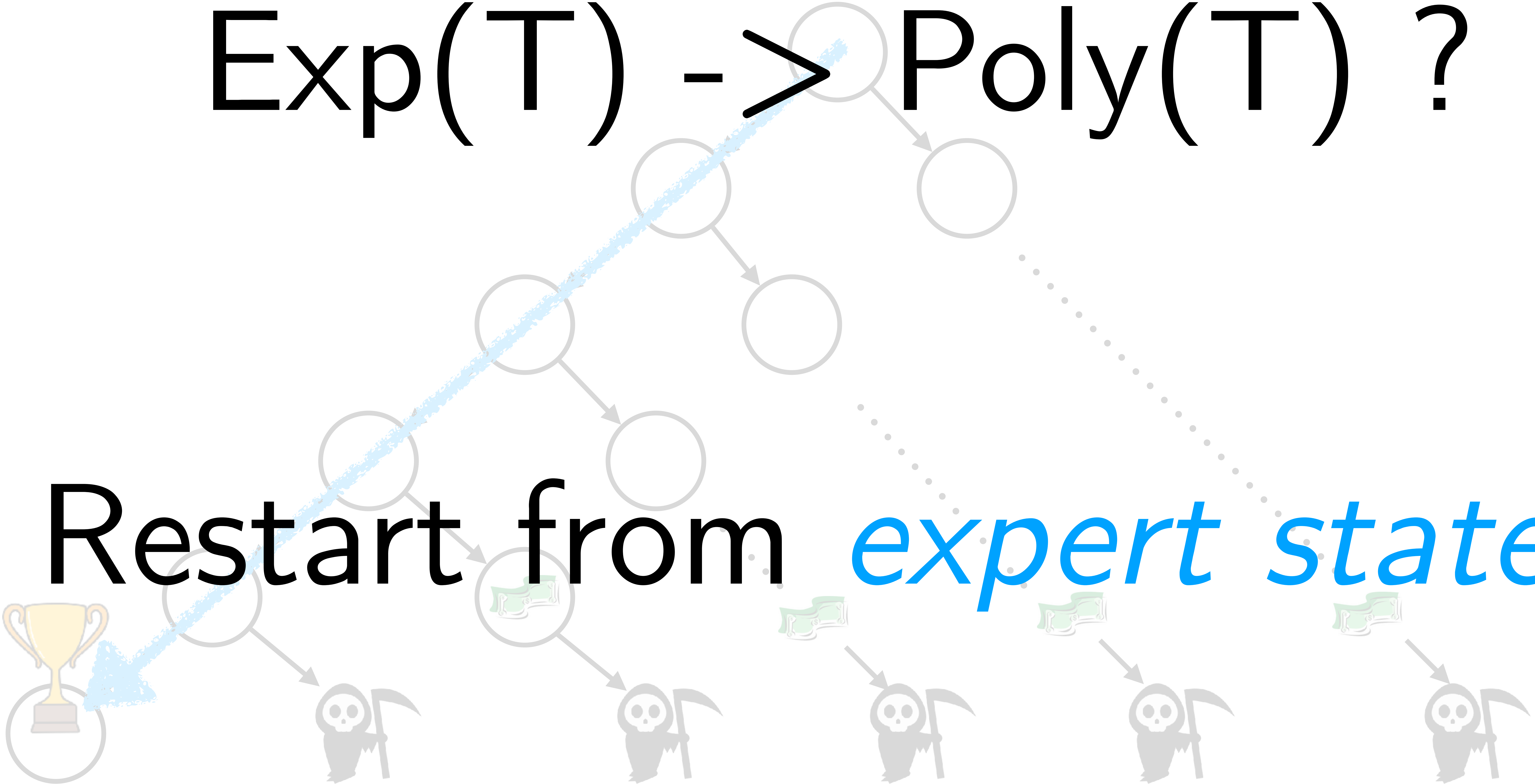
Anirudh Vemula [1] Yuda Song [2] Aarti Singh [2] J. Andrew Bagnell [1 2] Sanjiban Choudhury [3]

How do we turn planning Exp(T) -> Poly(T) ?



How do we turn planning Exp(T) -> Poly(T) ?



Restart from *expert states*

Policy Search via Dynamic Programming (PSDP)

(Bagnell, et al. 2003)

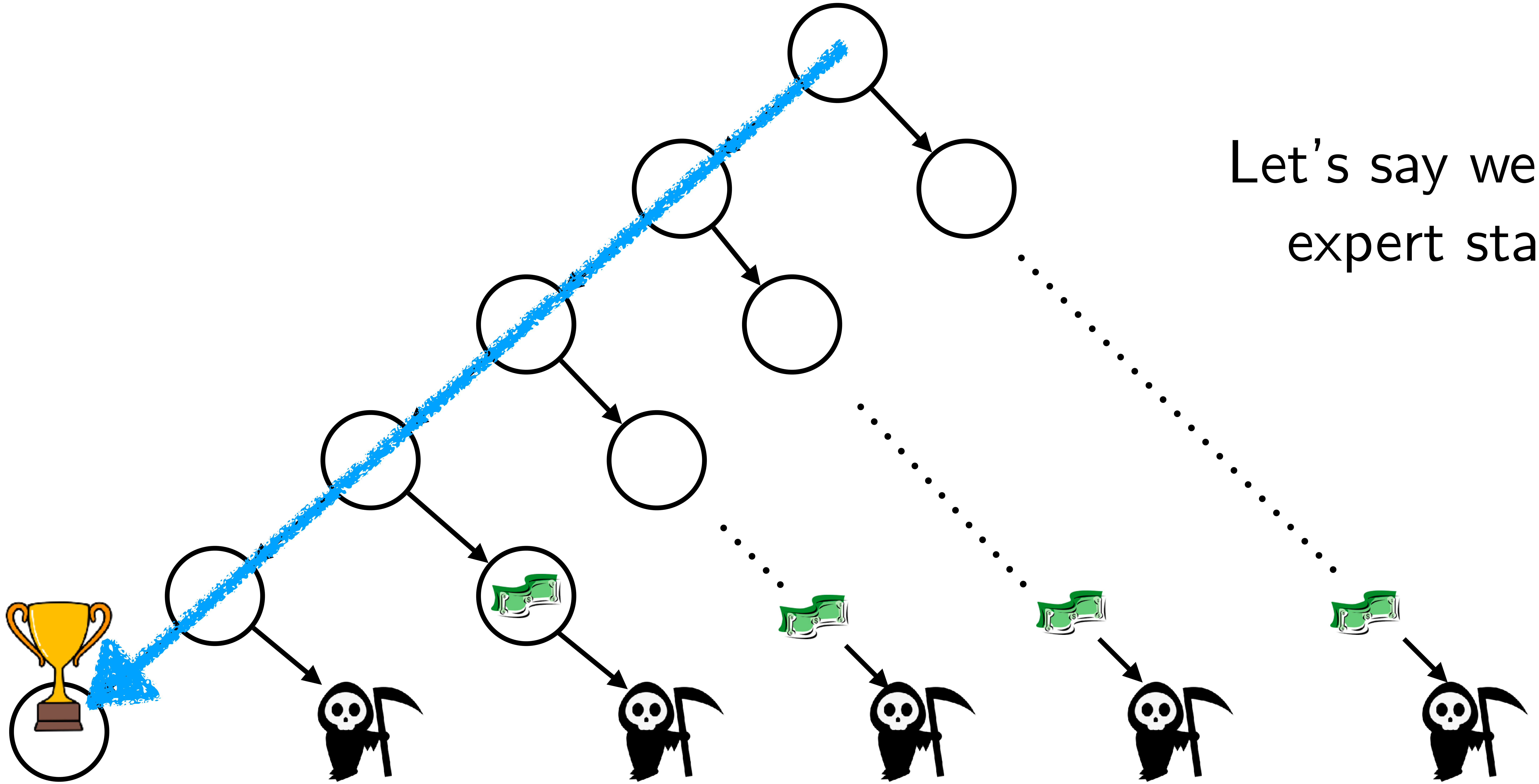
Iterate from T-1 and go back in time

At each time t, restart from expert state s_t^*

Solve for best policy π_t , *given future policies* $\pi_{t+1}, \pi_{t+2}, \cdots \pi_T$

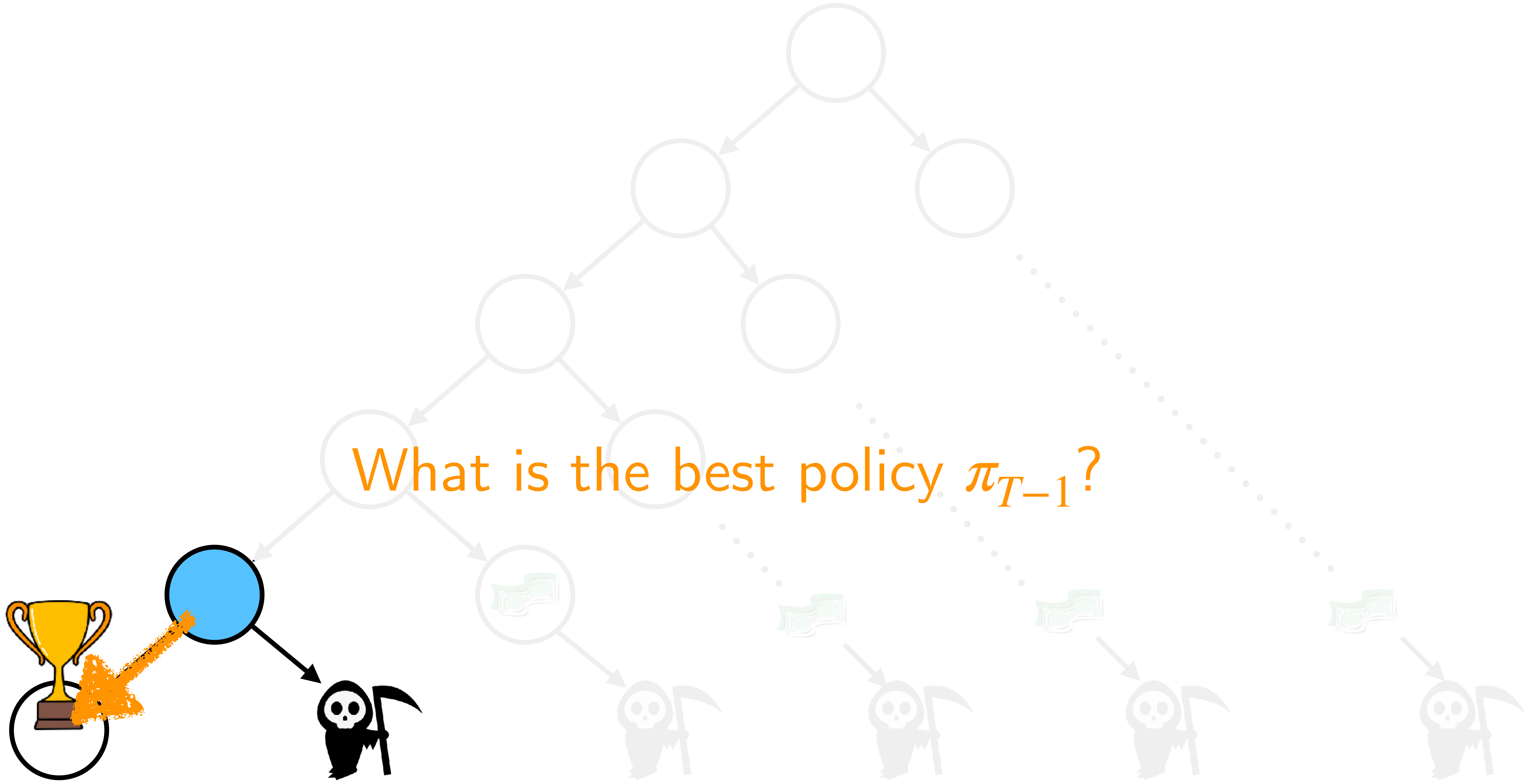
$$\pi_t = \arg\max_{\pi} r(s_t^*, \pi(s_t^*)) + \mathbb{E}_{s_{t+1}} V^{\pi_{t+1:T}}(s_{t+1})$$

Policy Search via Dynamic Programming (PSDP)



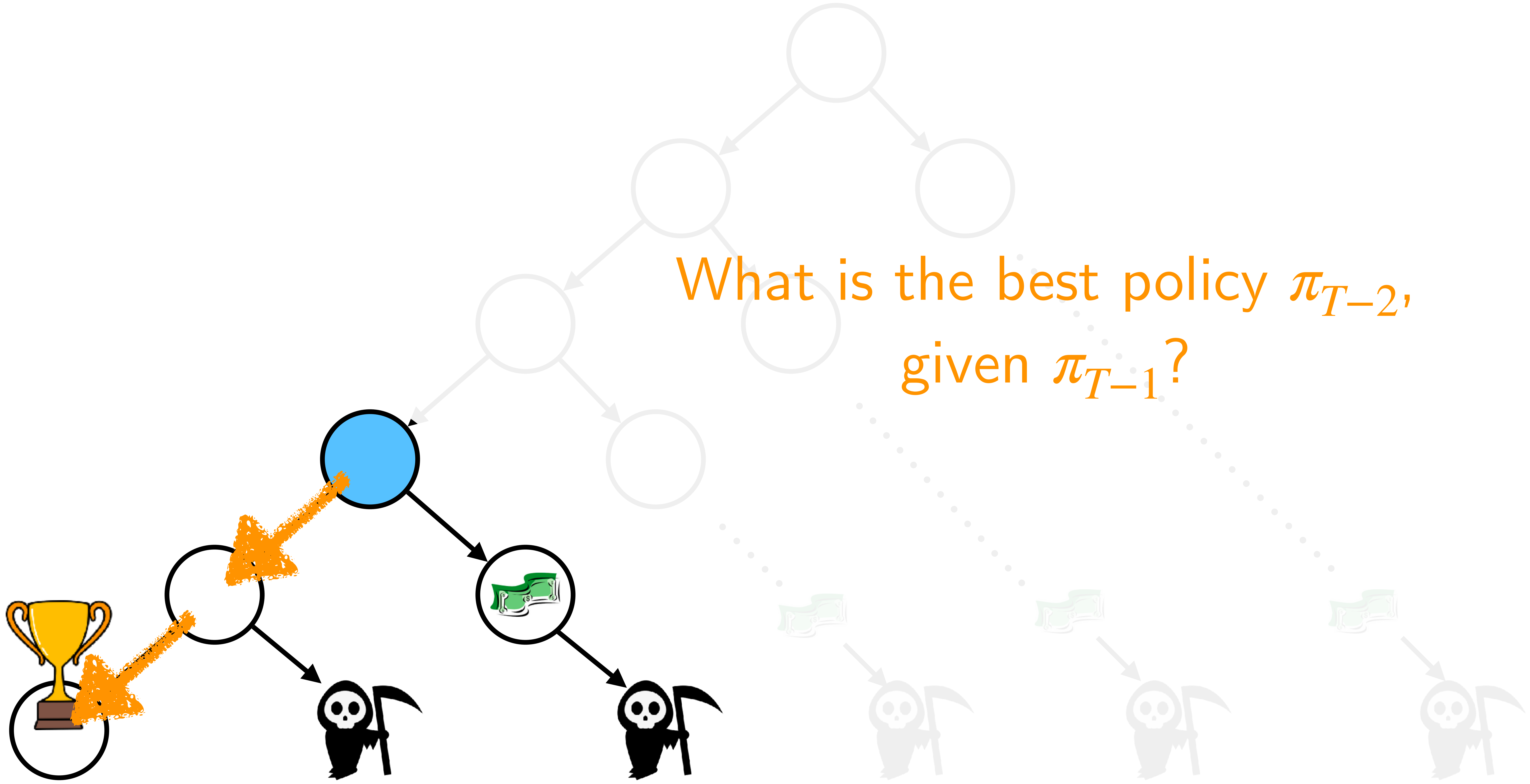
Let's say we have expert states

Policy Search via Dynamic Programming (PSDP)



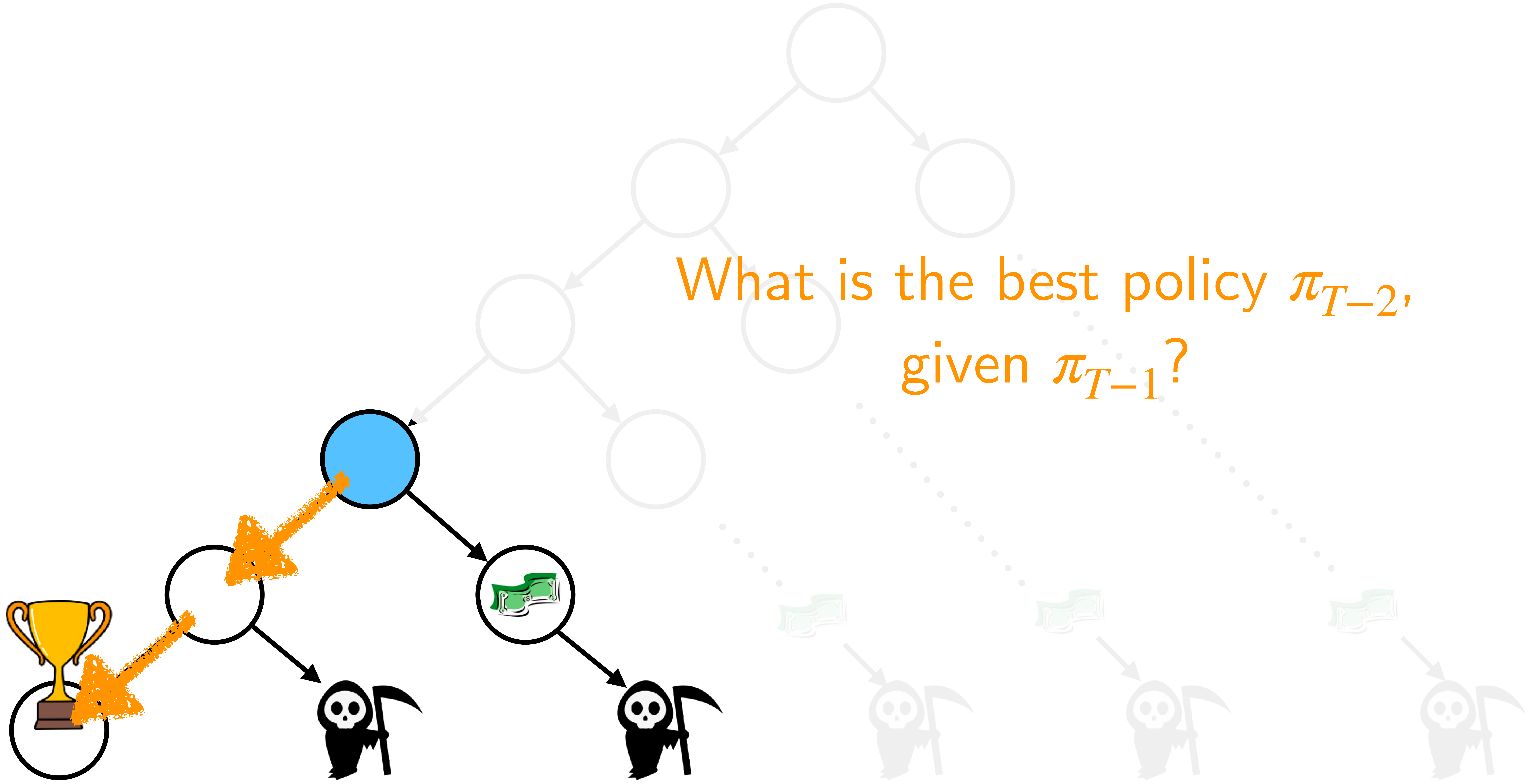
What is the best policy π_{T-1}?

Policy Search via Dynamic Programming (PSDP)



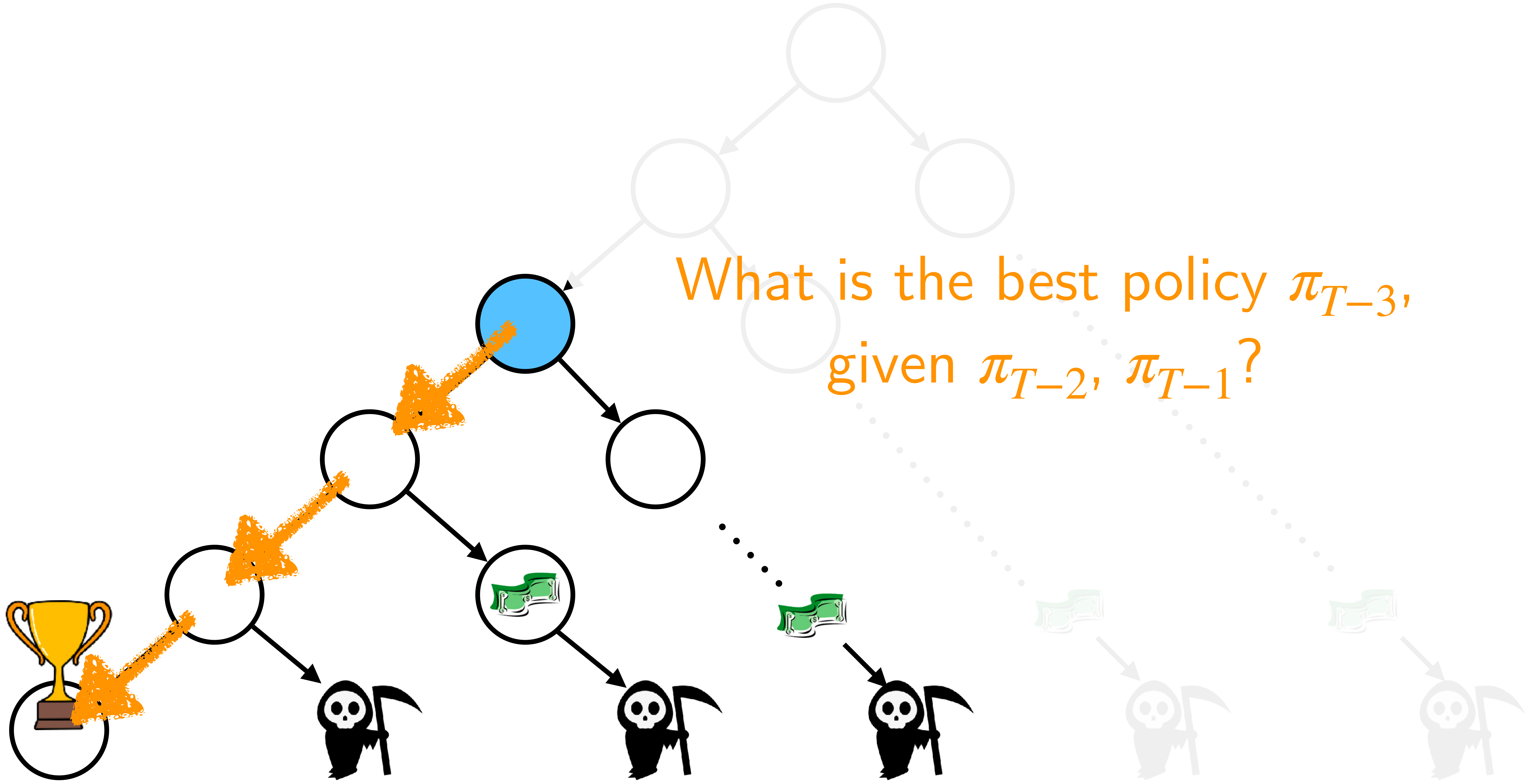
What is the best policy π_{T-2},
given π_{T-1}?

Policy Search via Dynamic Programming (PSDP)



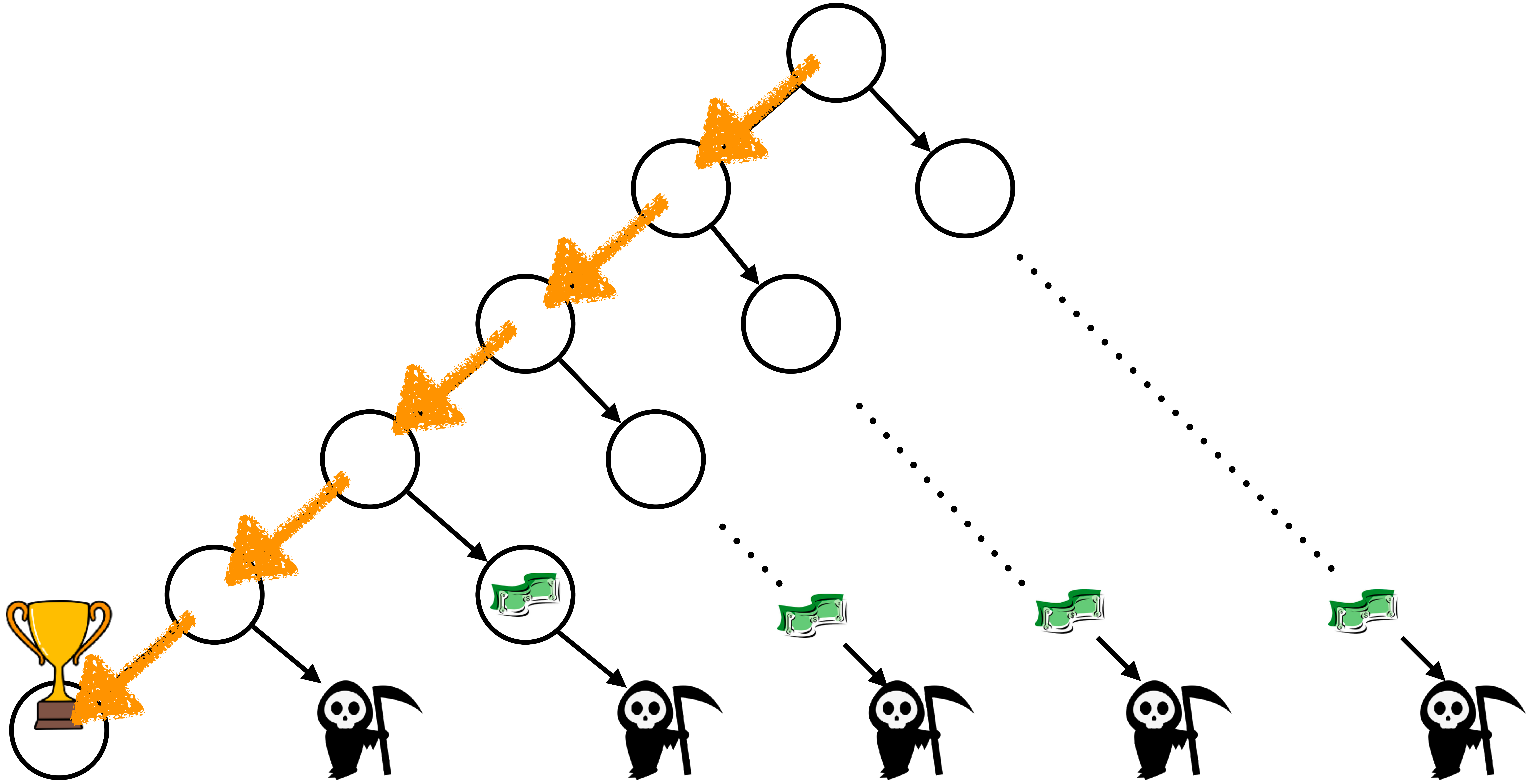
What is the best policy π_{T-2}, given π_{T-1}?

Policy Search via Dynamic Programming (PSDP)

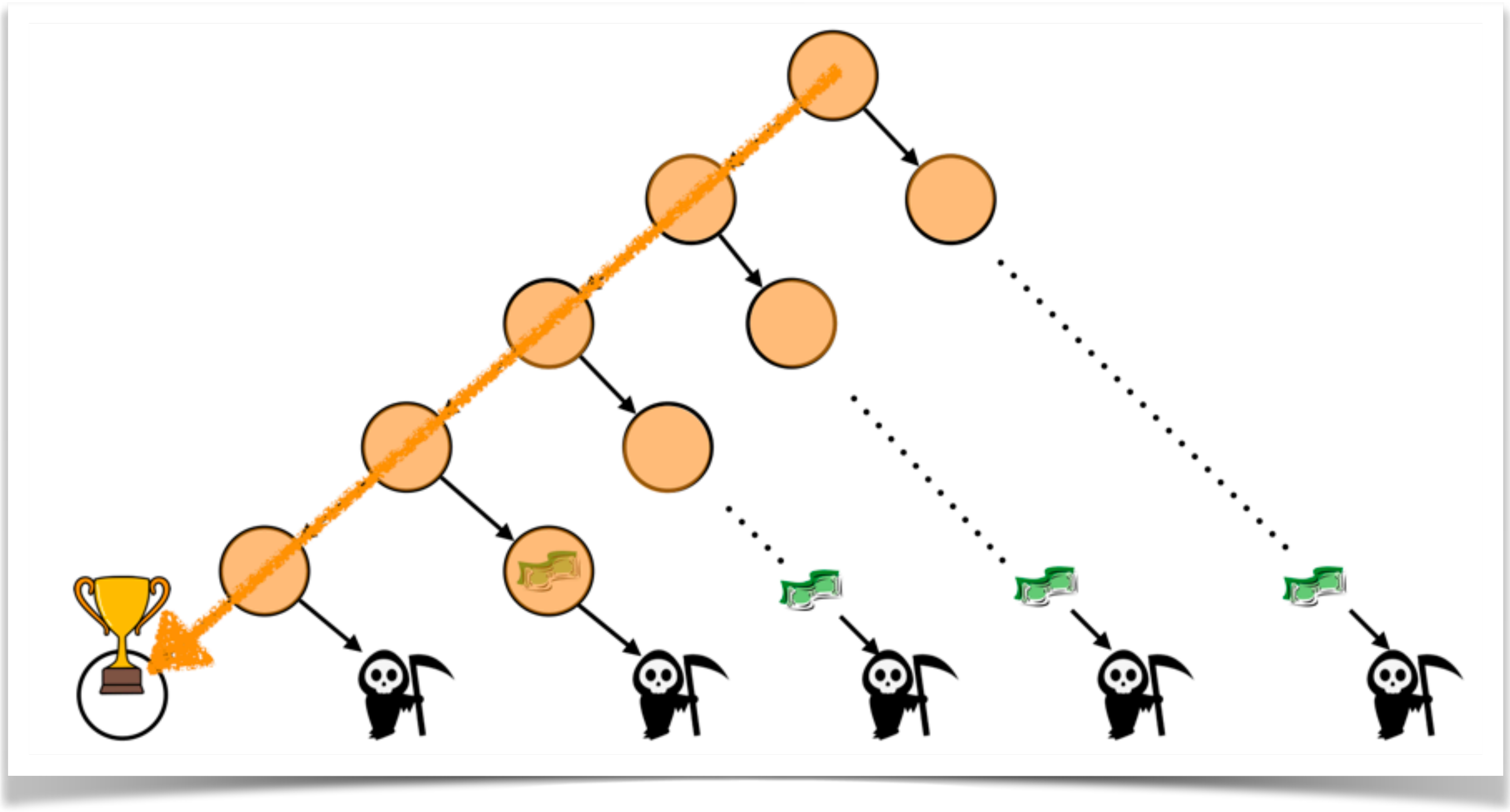


What is the best policy π_{T-3},
given π_{T-2}, π_{T-1}?

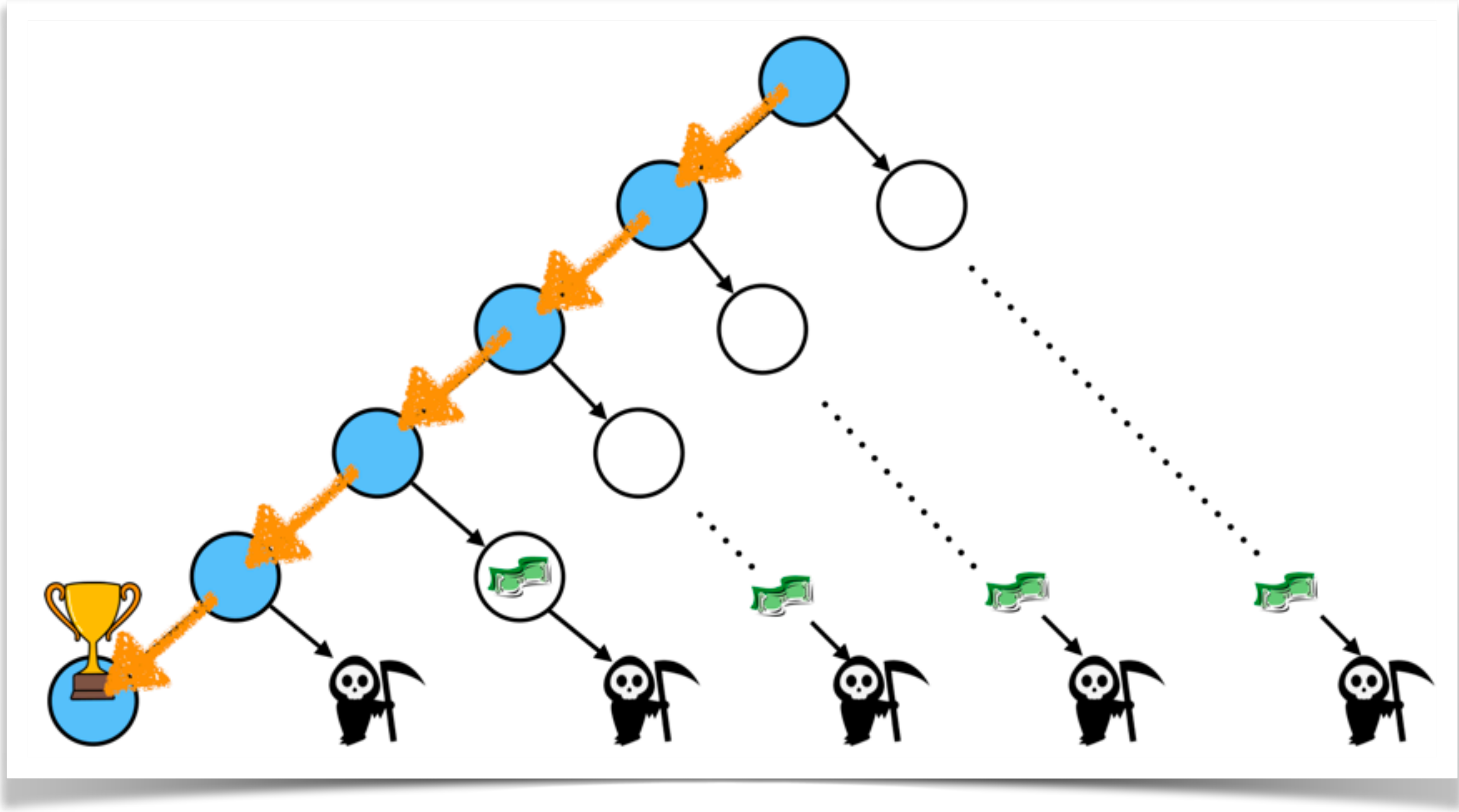
Only took poly(T) steps!



PSDP is Lazy



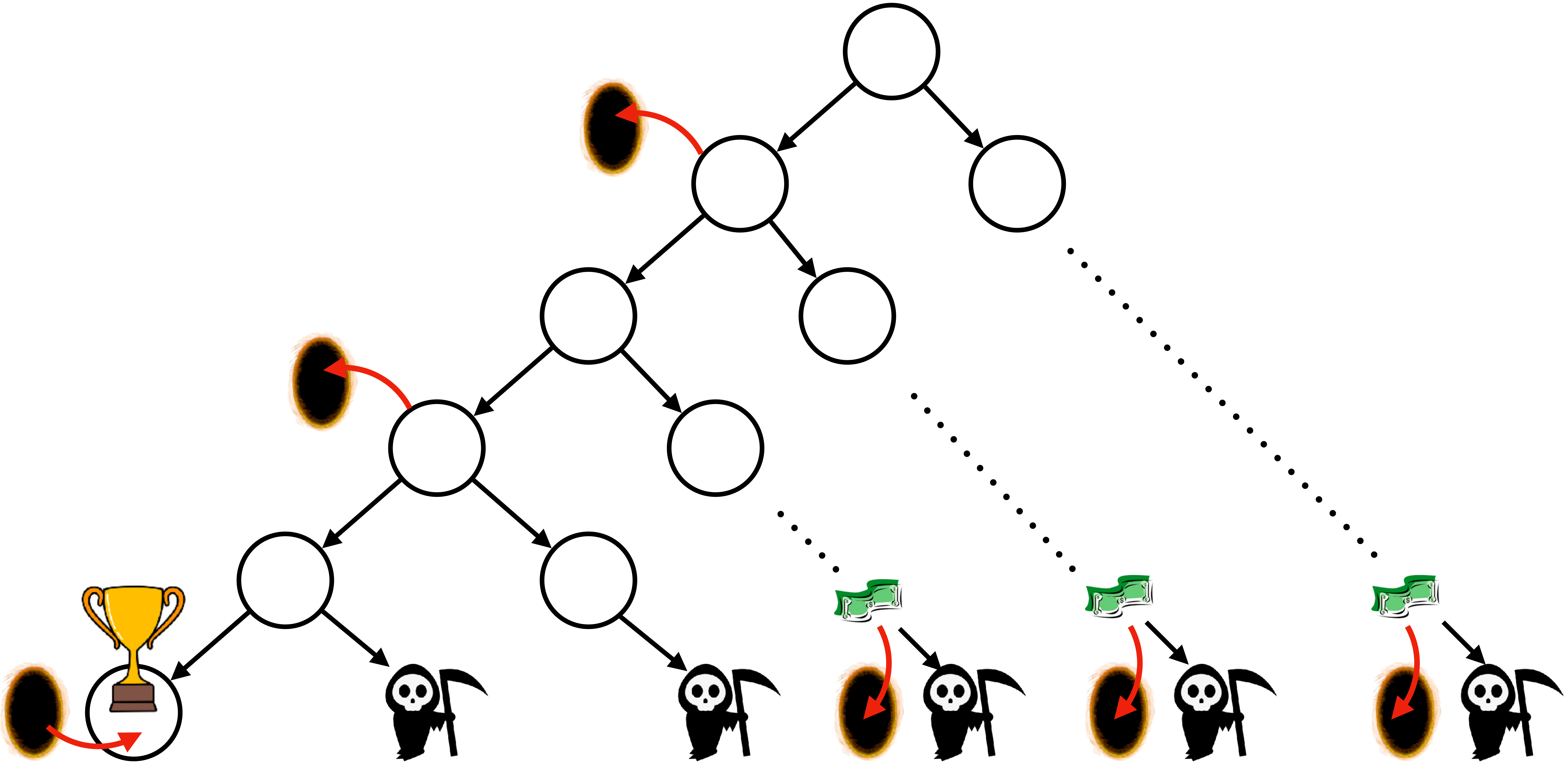
Instead of searching all states
to find the best policy



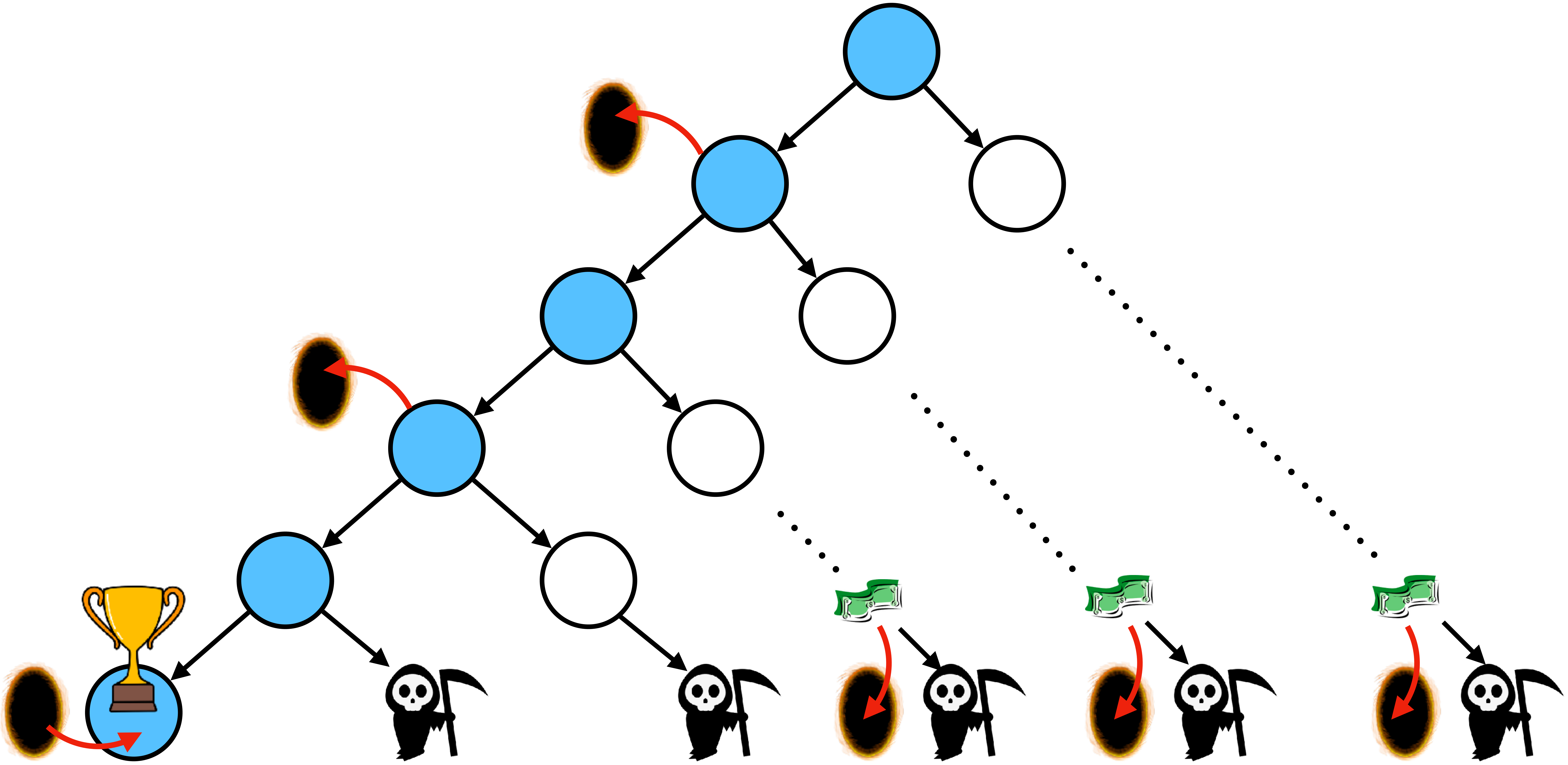
Just do better on states
the expert visits

Is being lazy
a good idea
for model learning?

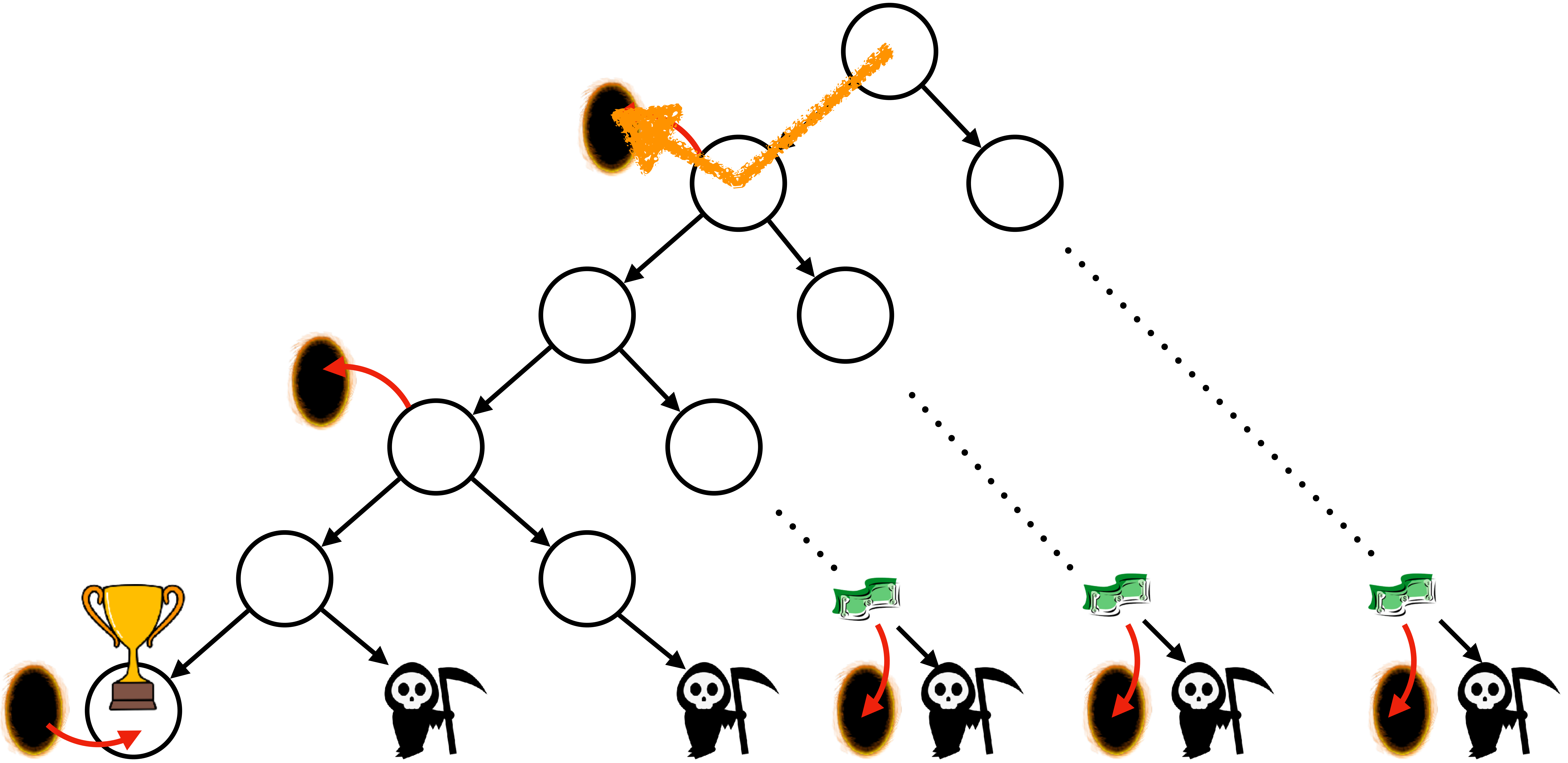
Model at iteration 0



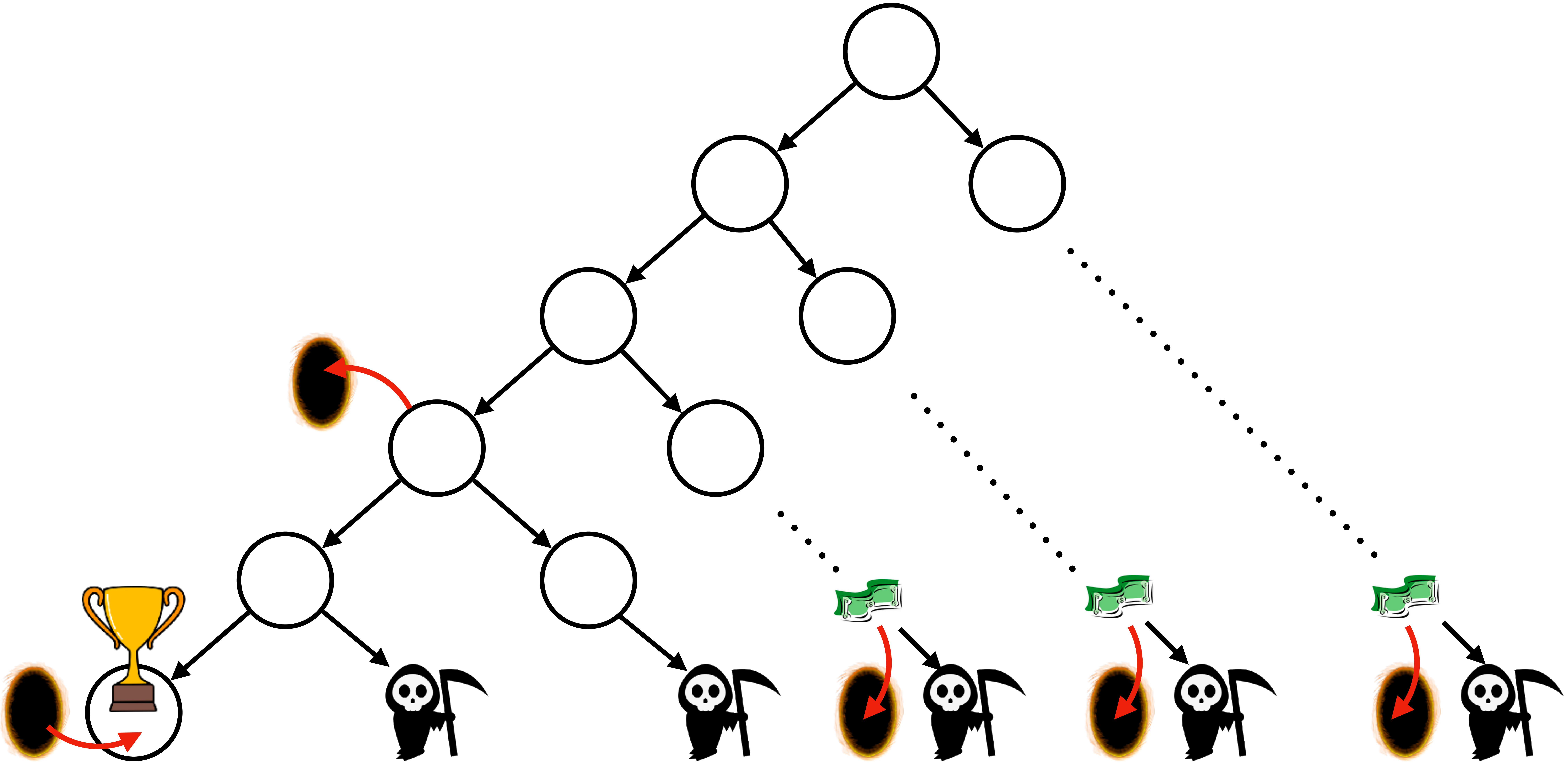
Run lazy policy search poly(T)



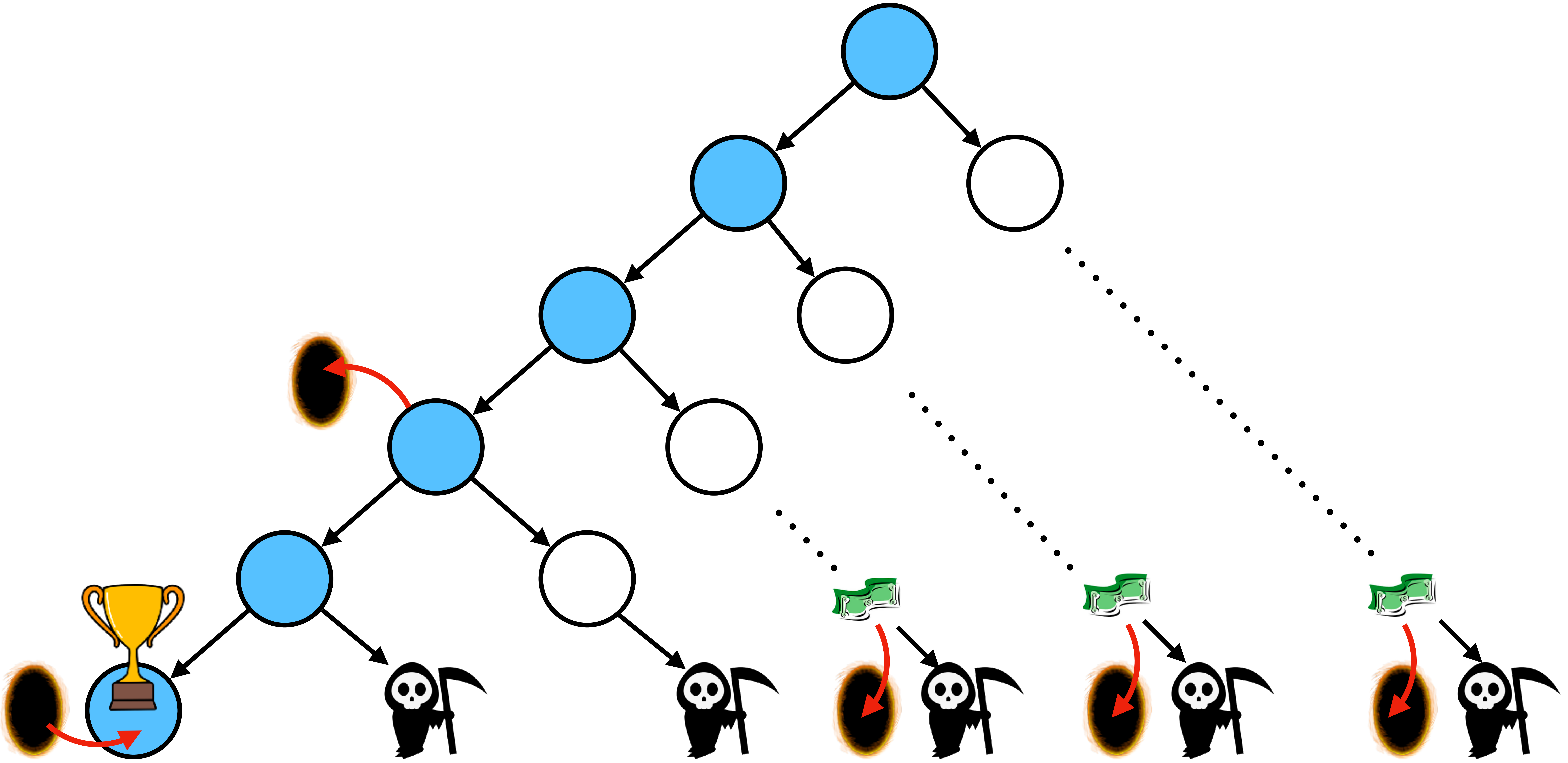
Policy at iteration 0



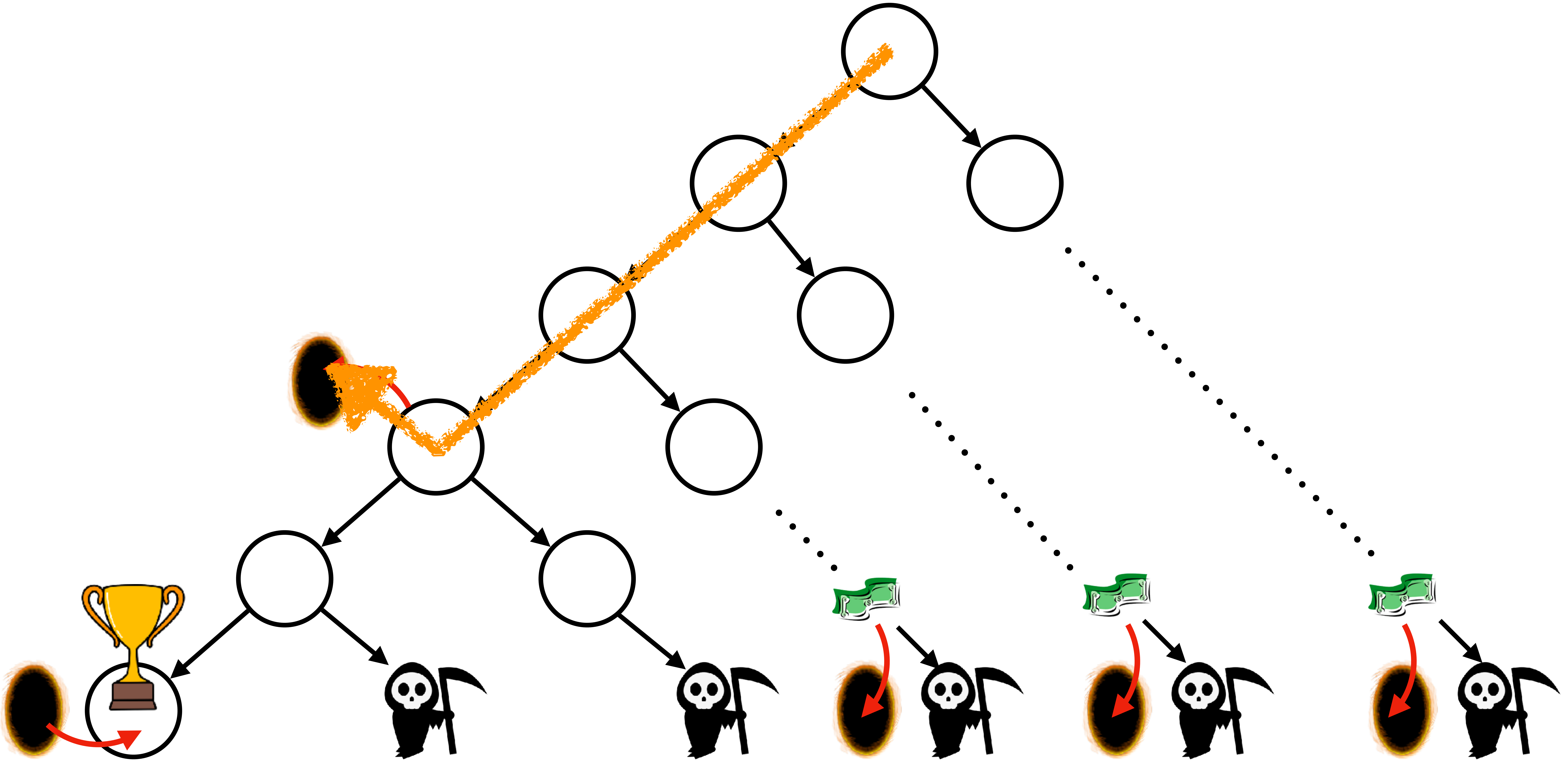
Model at iteration 1



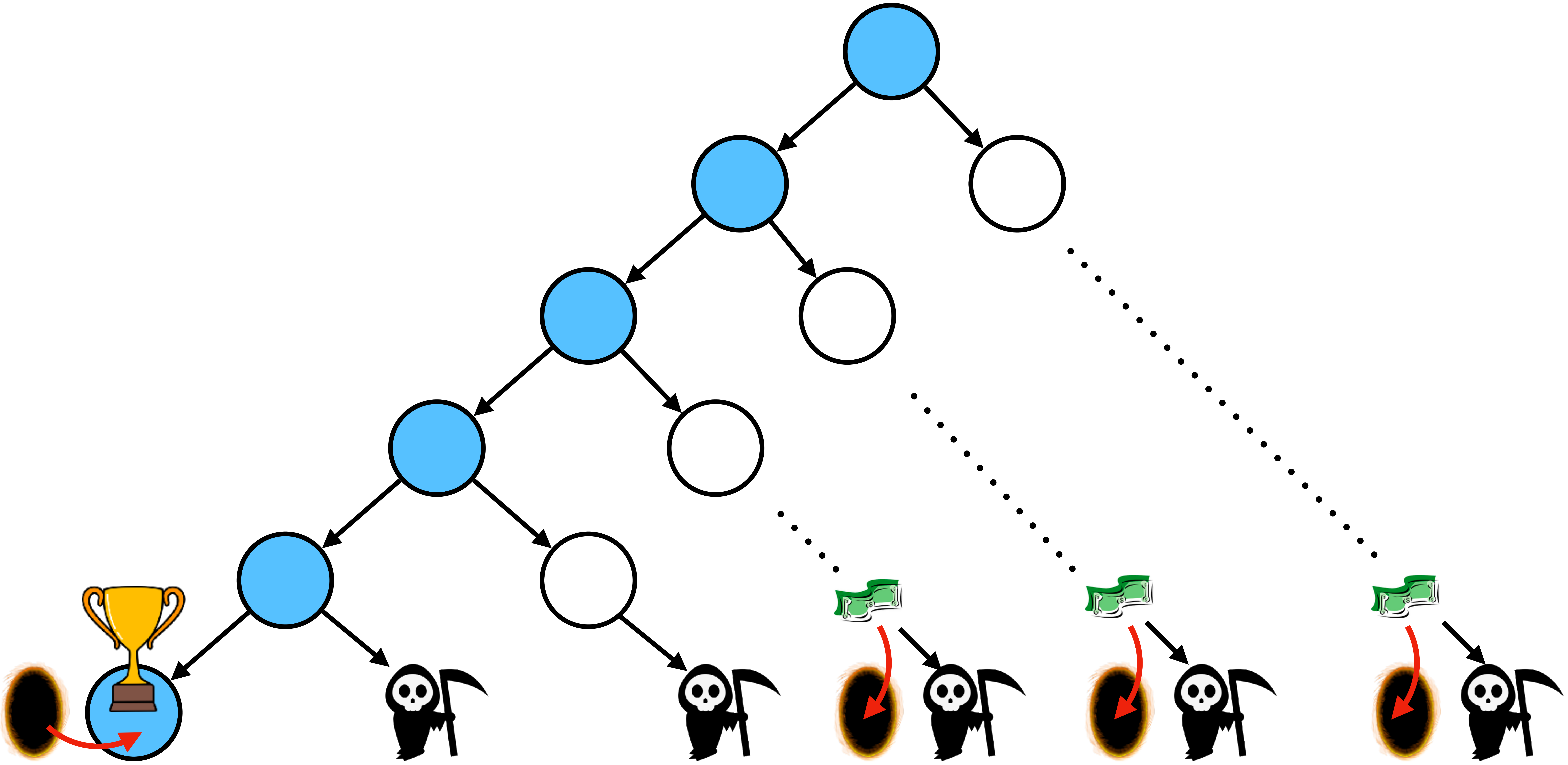
Run lazy policy search poly(T)



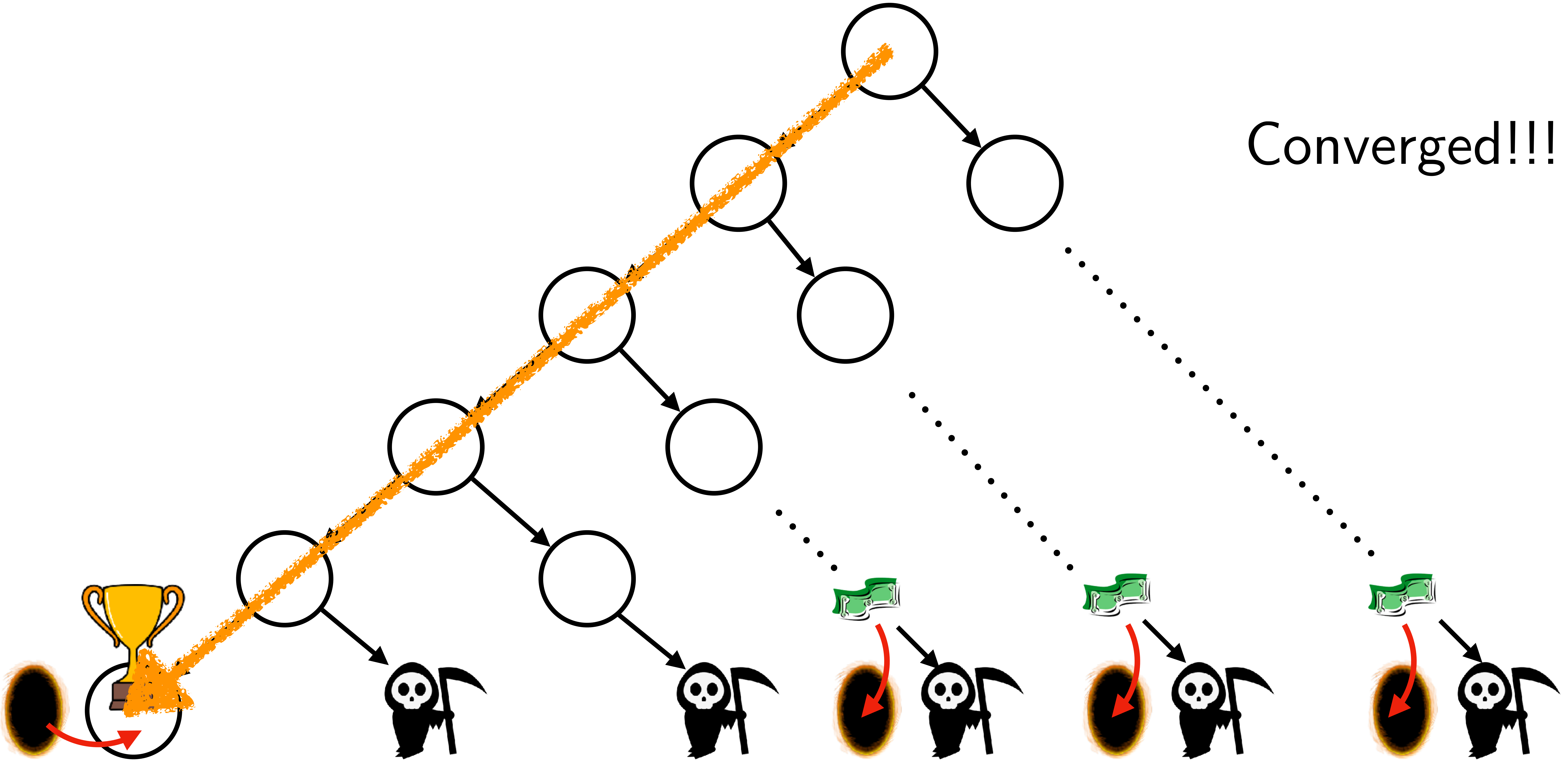
Policy at iteration 1



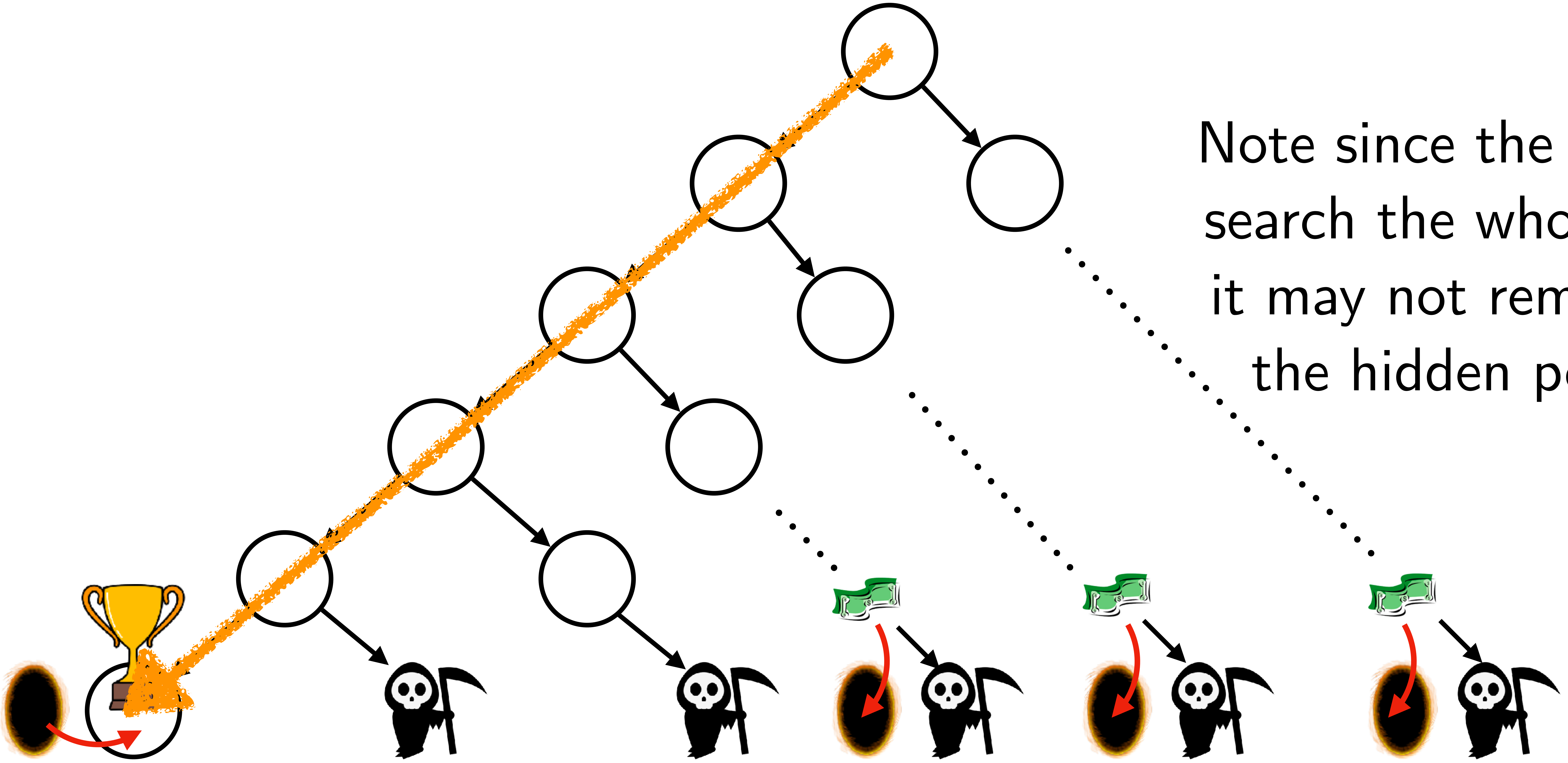
Run lazy policy search poly(T)



Policy at iteration 2



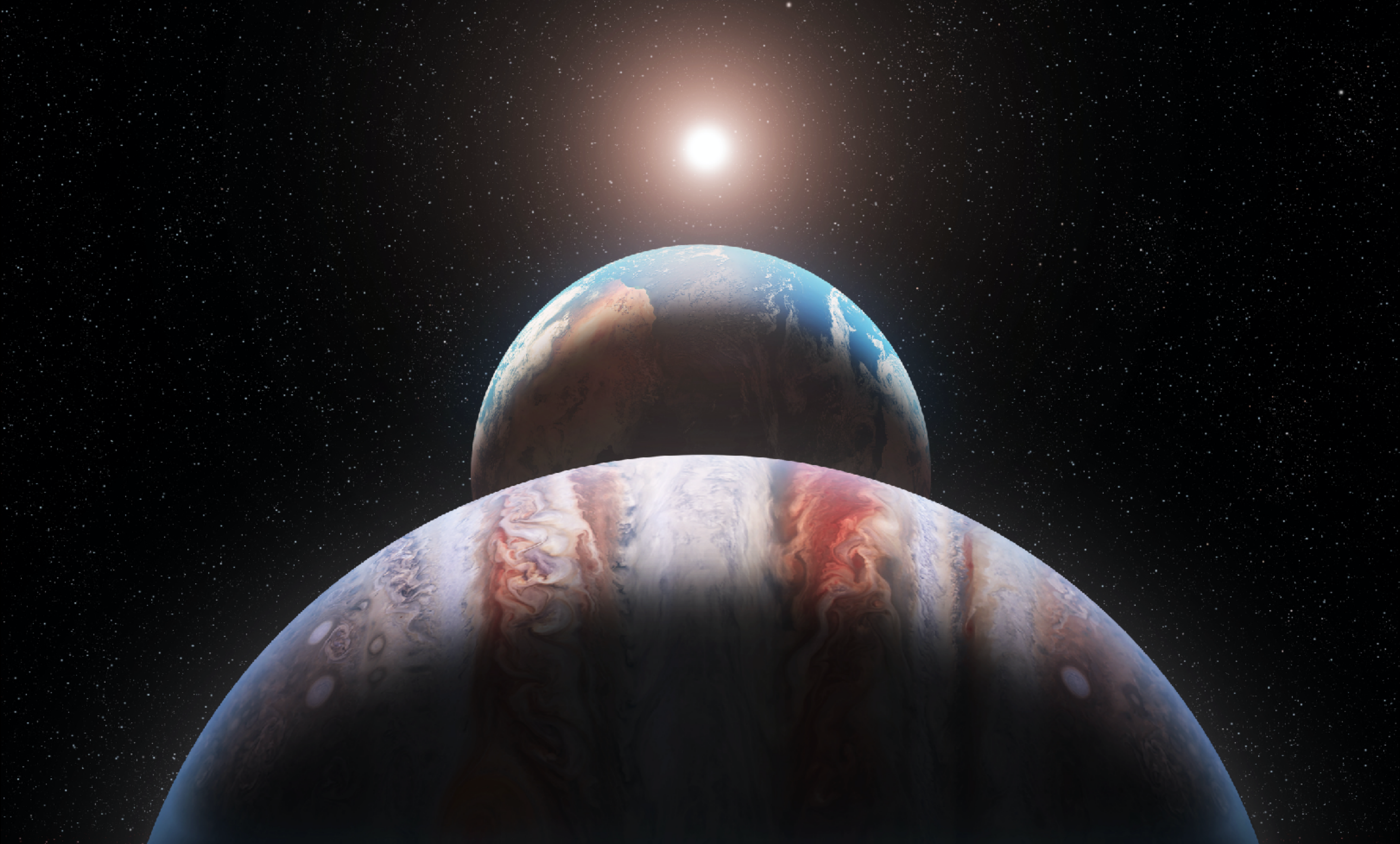
Final Model + Policy



Note since the planner search the whole tree, it may not remove all the hidden portals

But can we prove that lazy is good for model learning?

A New Lemma!



Lemma: Performance Difference via Advantage in Model

$$J_{M^*}(\pi^*) - J_{M^*}(\hat{\pi})$$

$$\leq \mathbb{E}_{s^* \sim \pi^*}\left[A^{\pi}(s^*, a^*)\right] \quad + TV_{\max}\mathbb{E}_{s,a \sim \pi^*}||\hat{M}(s,a) - M(s,a)||$$

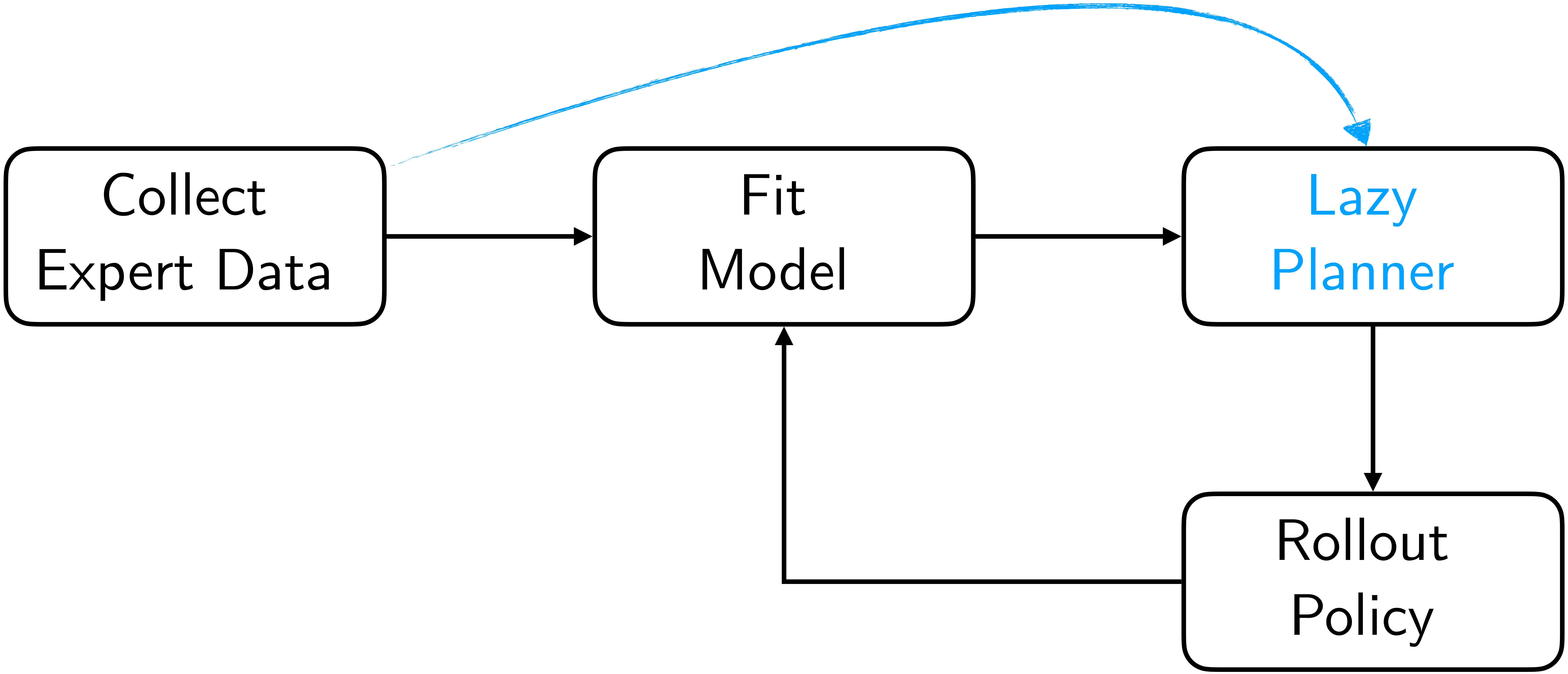
Advantage of expert in model

Model fit on expert states

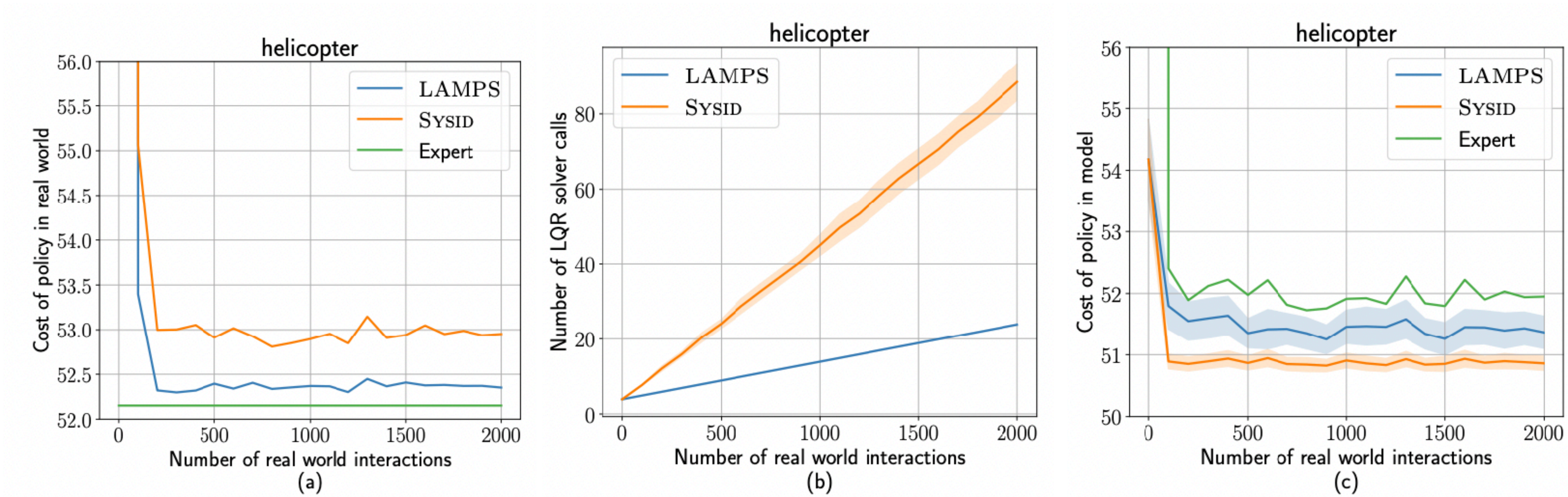
$$+ TV_{\max}\mathbb{E}_{s,a \sim \pi}||\hat{M}(s,a) - M(s,a)||$$

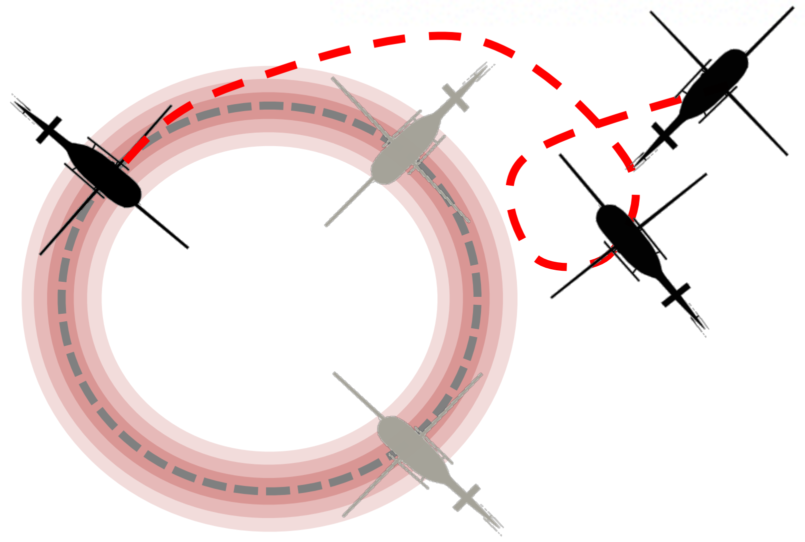
Model fit on policy states

Lazy Model-based Policy Search (LAMPS)



LAMPS finds a better policy with fewer samples + fewer computation

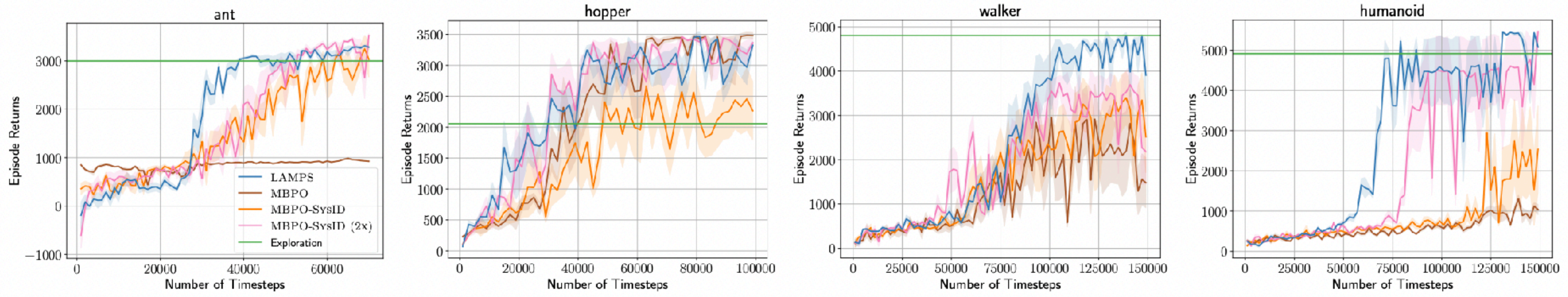




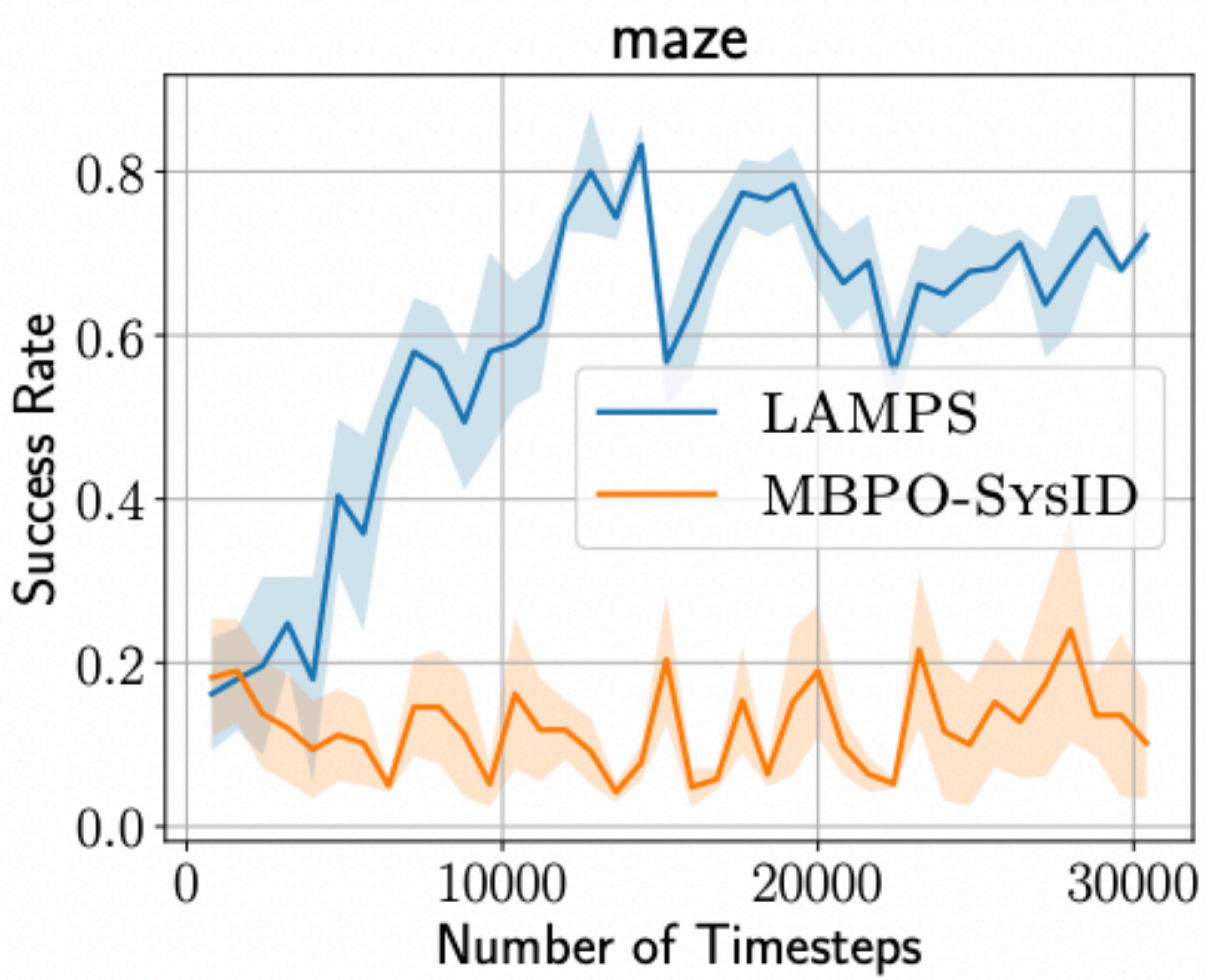
SysID: Use planner (iLQR)

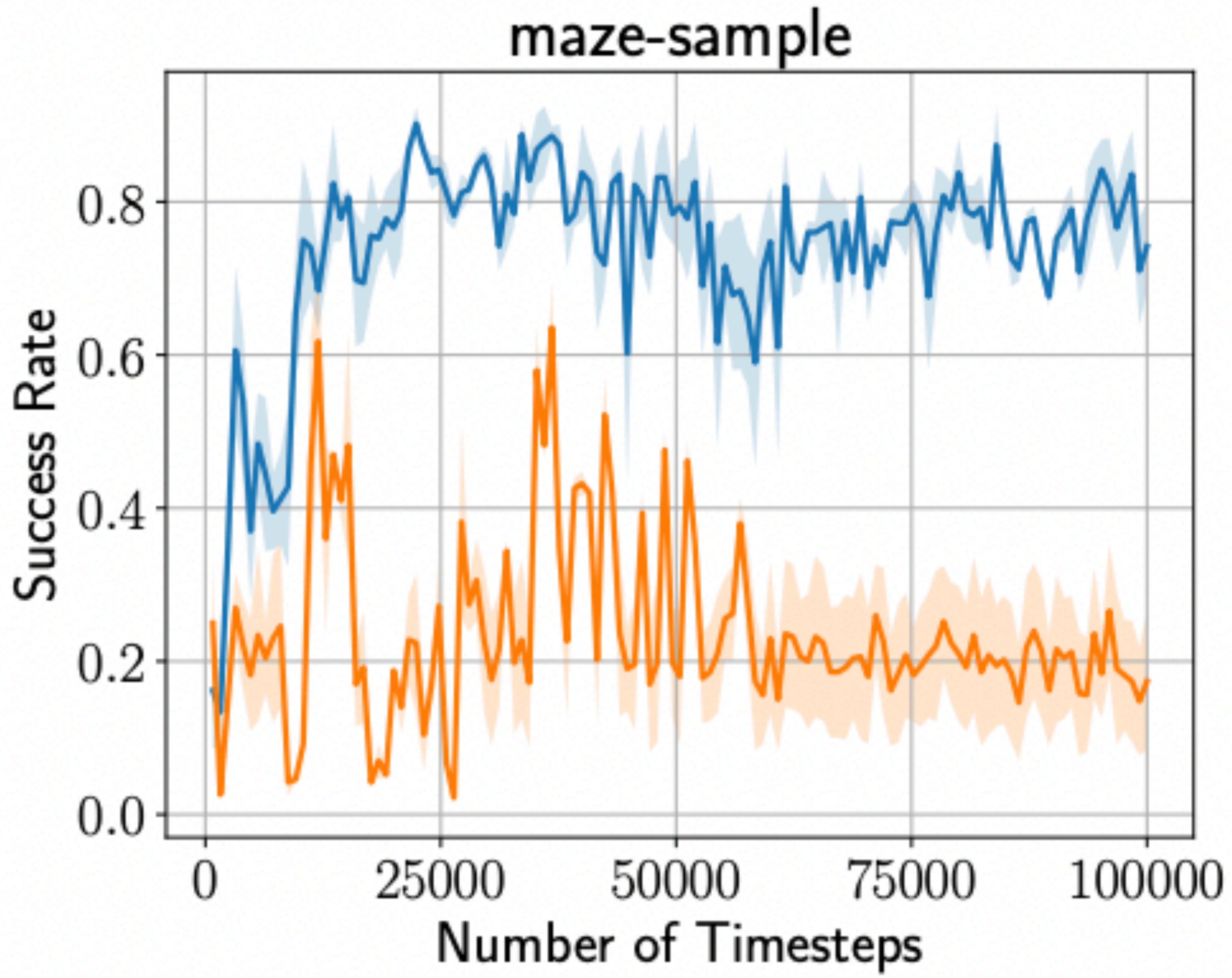
LAMPS: Use PSDP (LQR on expert traj)

LAMPS converges faster than both SysID and MBPO



LAMPS makes better use of Expert Data

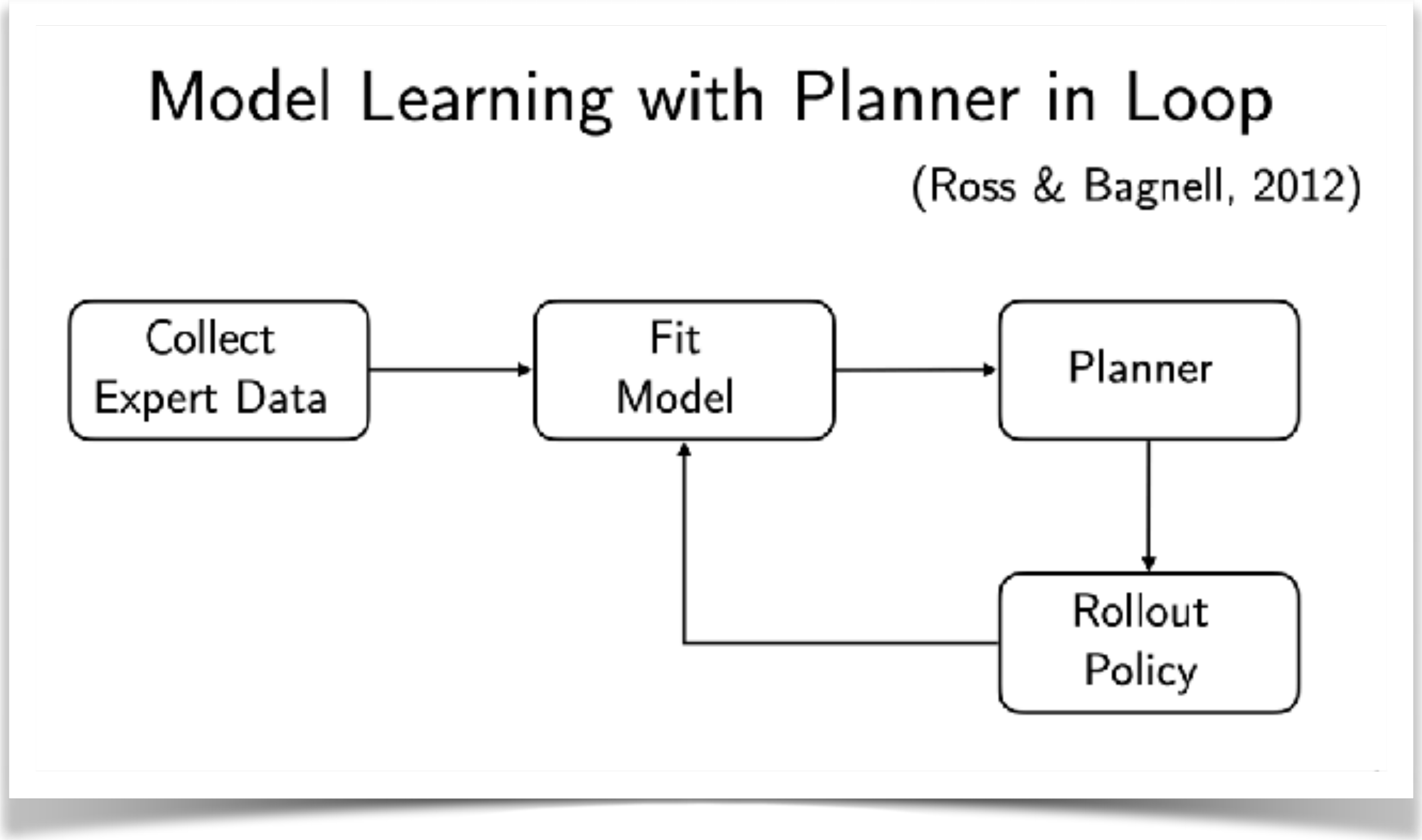




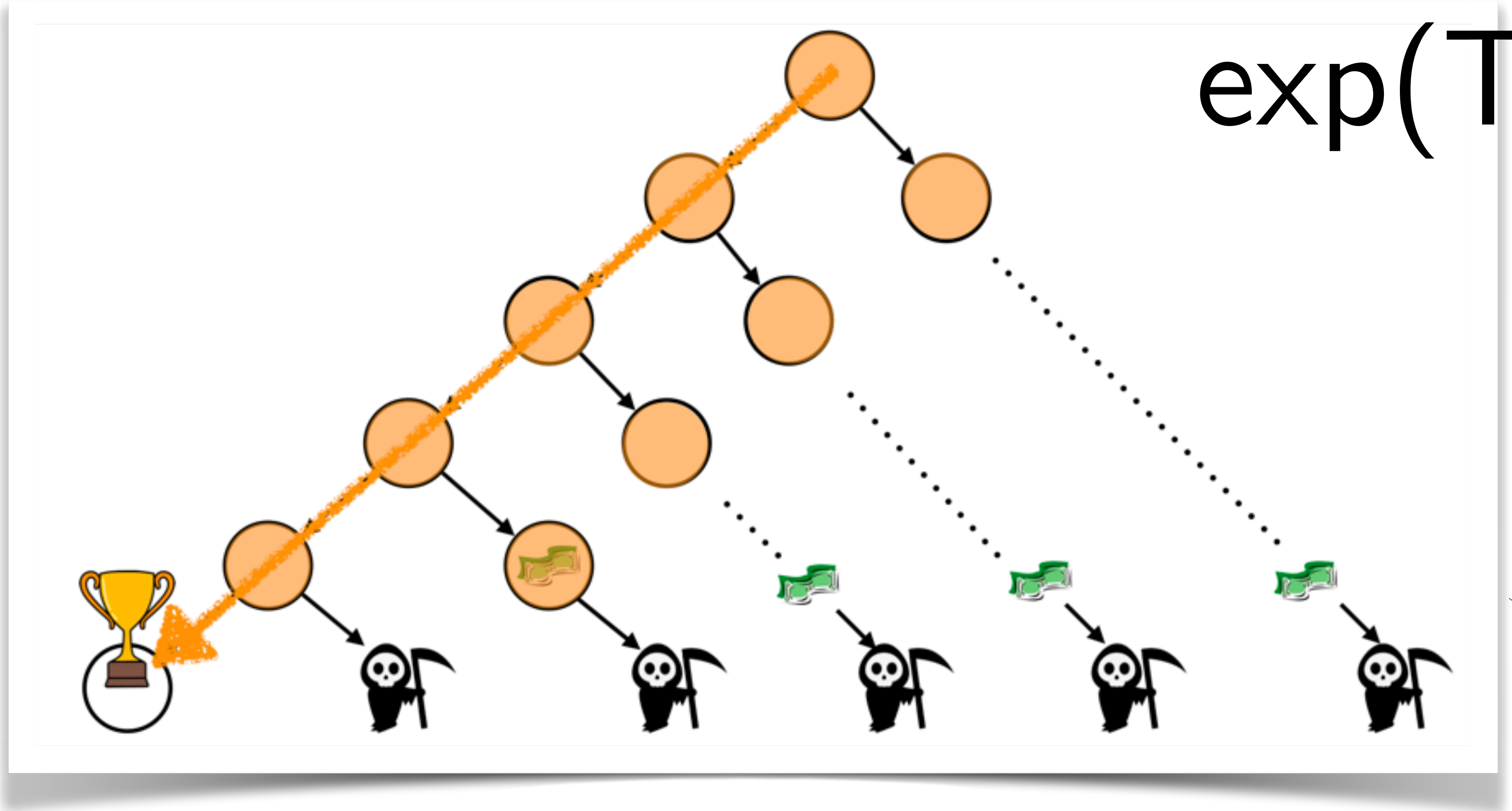
10000 samples

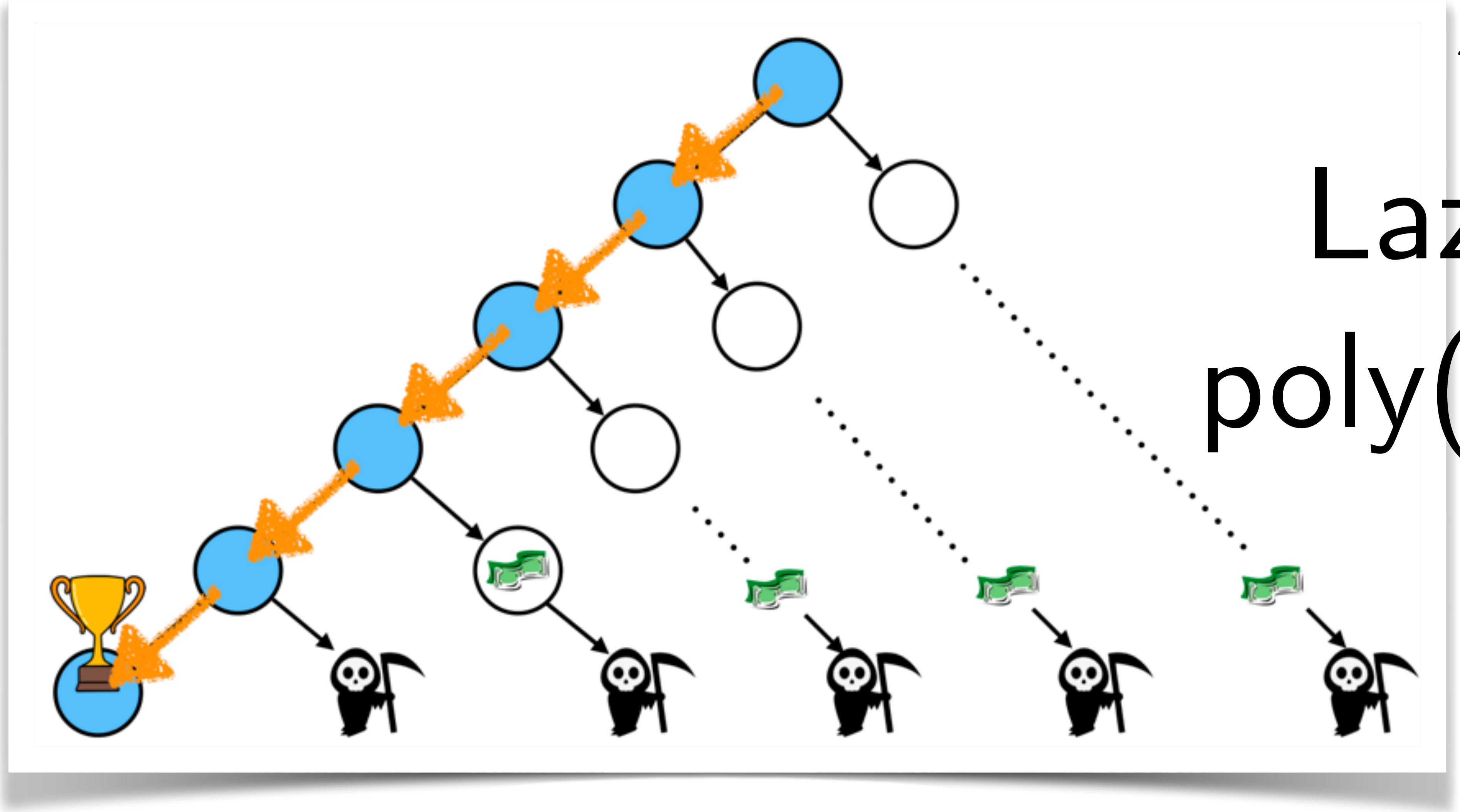
50000 samples

Recap

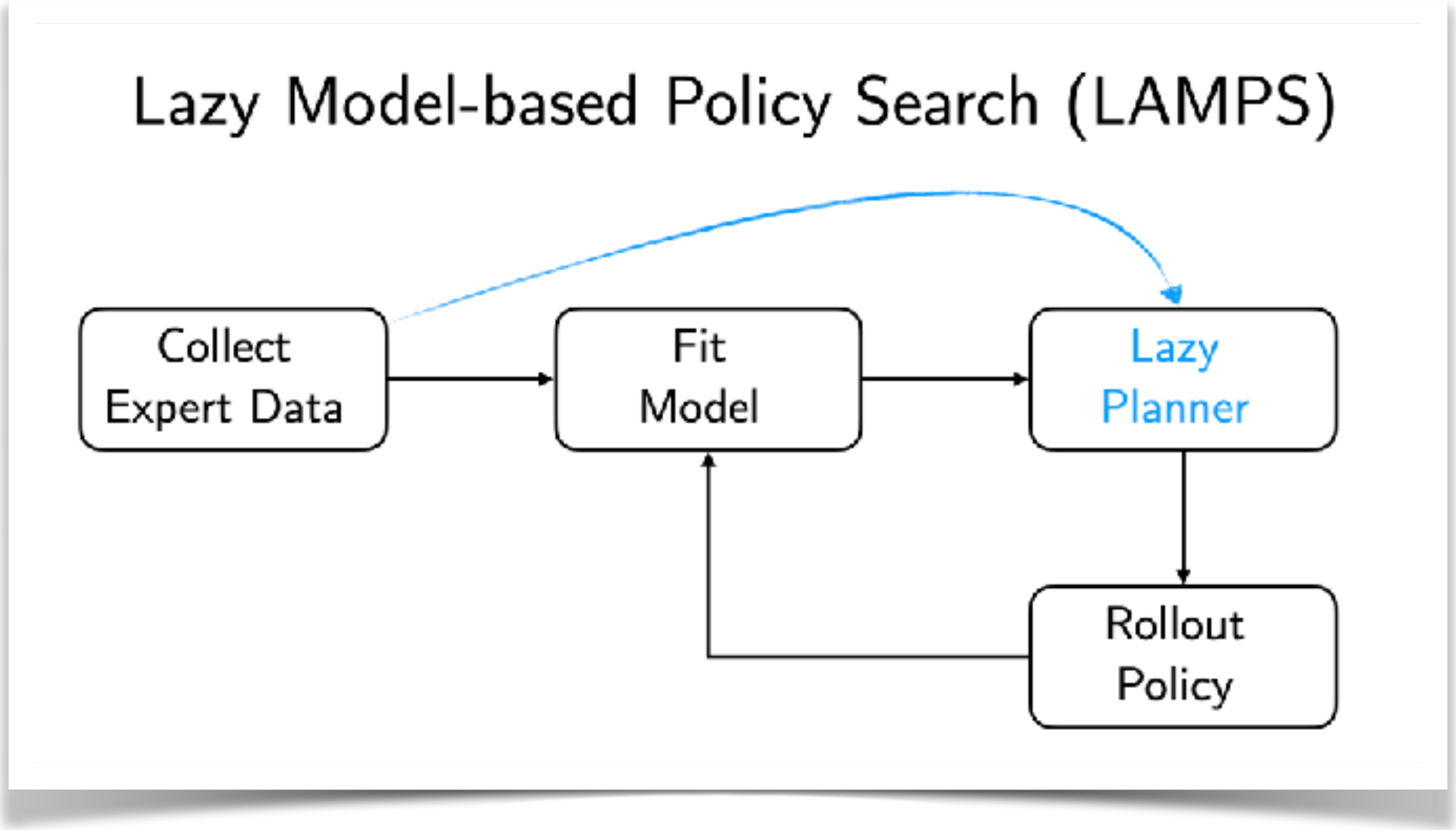


Planning exp(T)!





Lazy poly(T)!

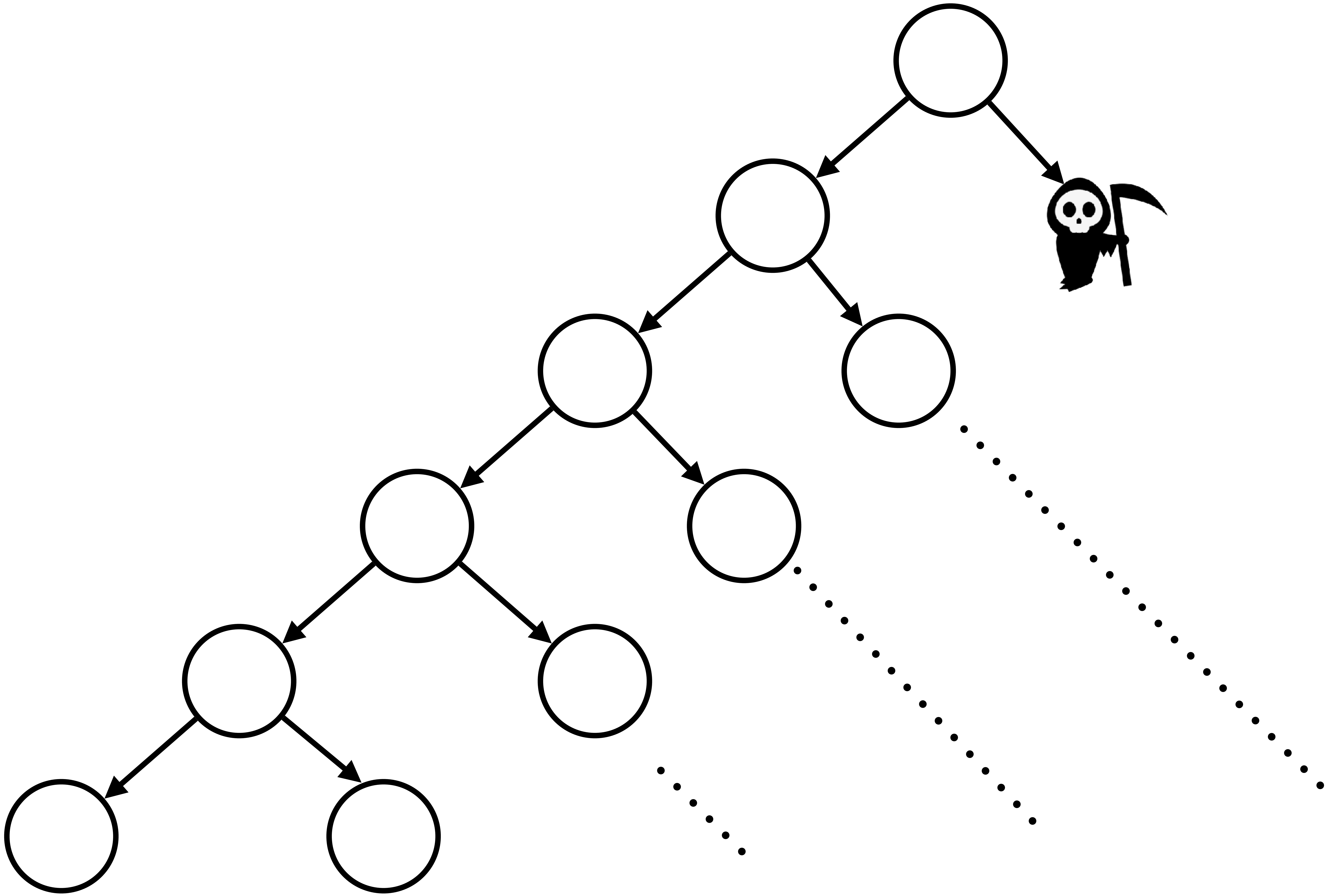


Another challenge.

Mismatched Objectives

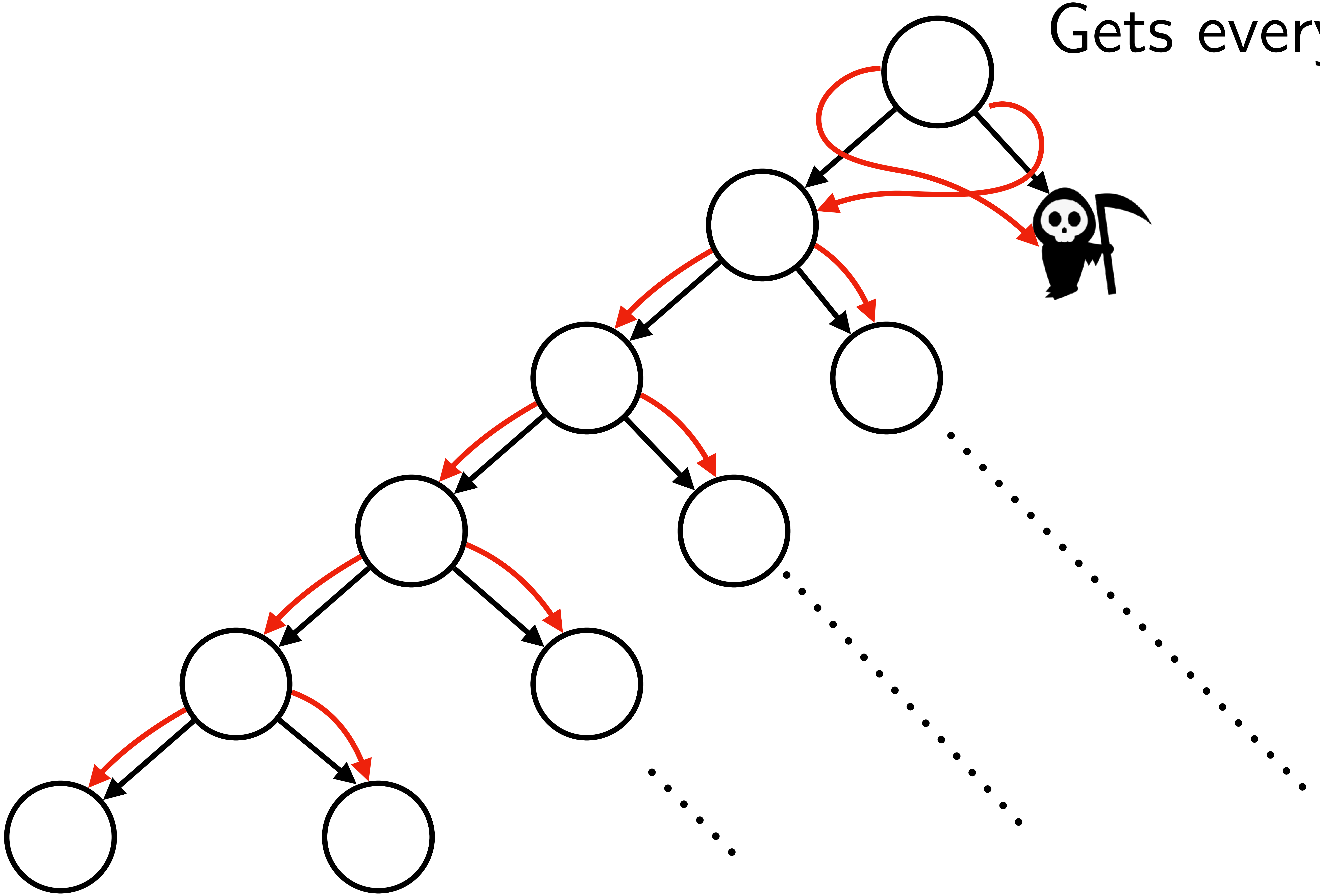
Fitting model with L2 loss
is mismatched
with how good
the resulting policy is

True Dynamics

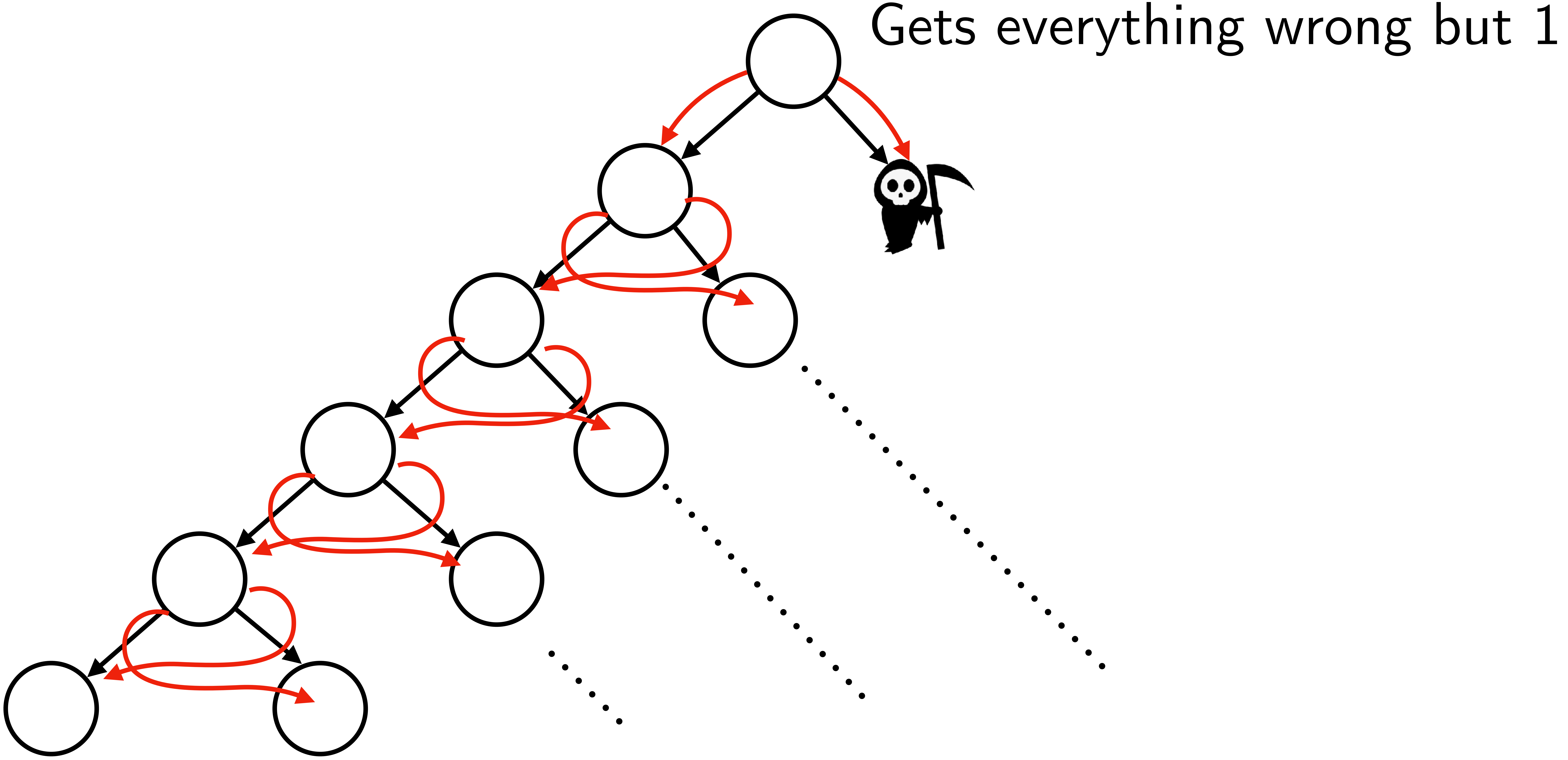


Learnt Model A

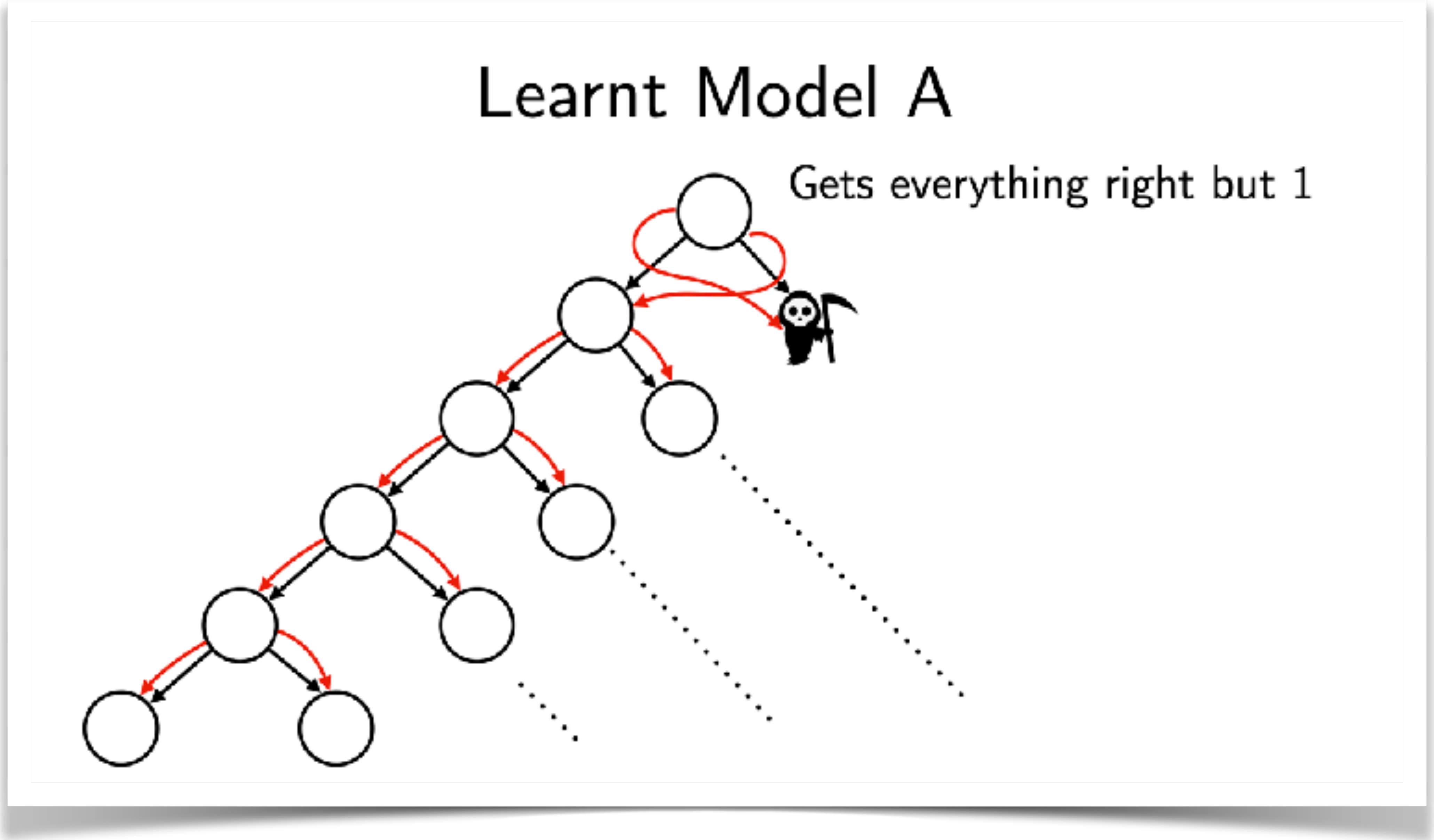
Gets everything right but 1

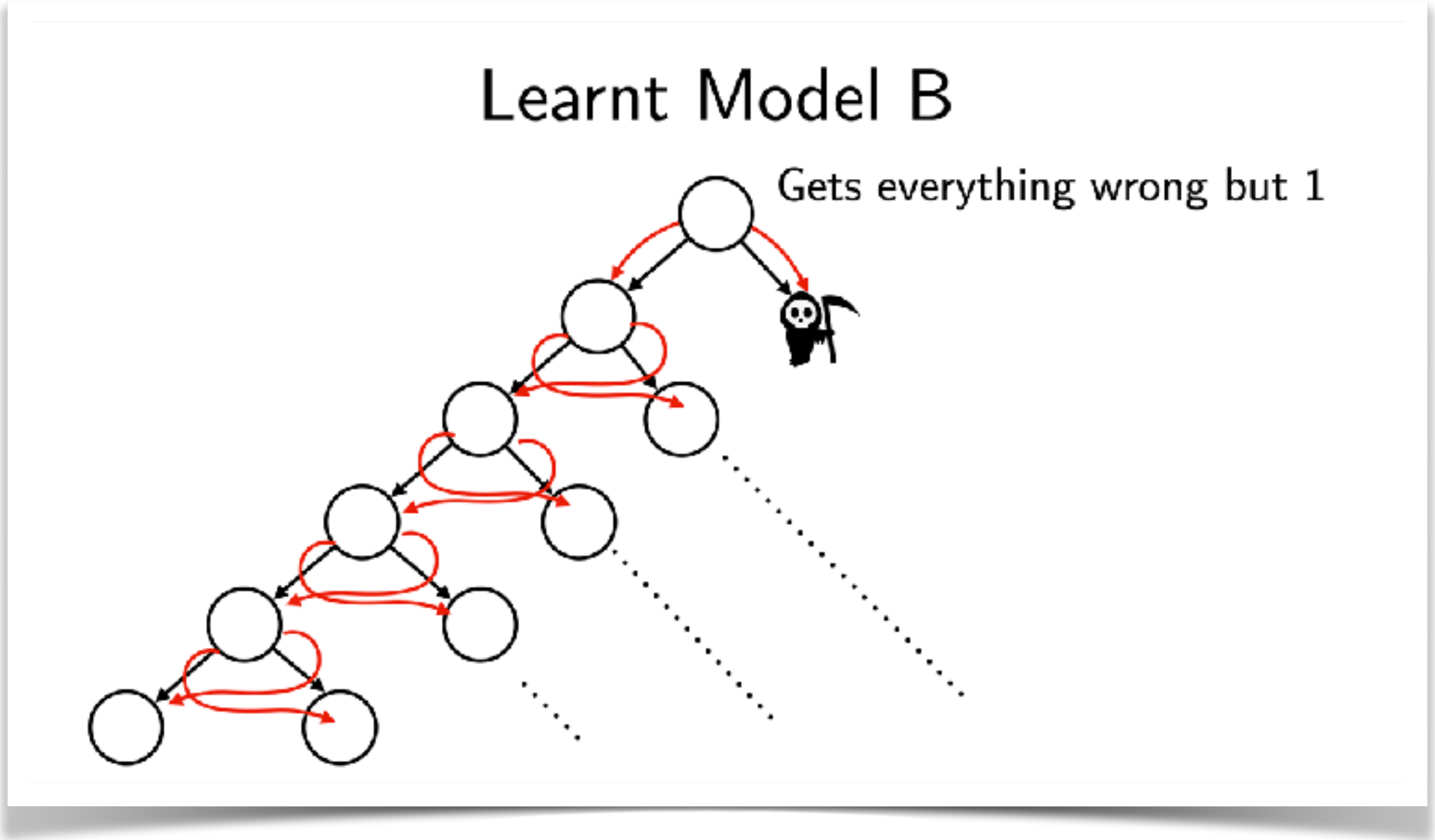


Learnt Model B



Which model has lower loss? Which one do we prefer?





Can we have change the loss for how we fit the model?

Our new lemma actually prescribes matching values!

$$J_{M^*}(\pi^*) - J_{M^*}(\hat{\pi})$$

$$= \mathbb{E}_{s^* \sim \pi^*}\left[A^{\hat{\pi}}(s^*, a^*)\right] \quad + T\mathbb{E}_{s,a \sim \pi^*}\left[E_{s' \sim \hat{M}} V^{\hat{\pi}}(s') - E_{s'' \sim M^*} V^{\hat{\pi}}(s'')\right]$$

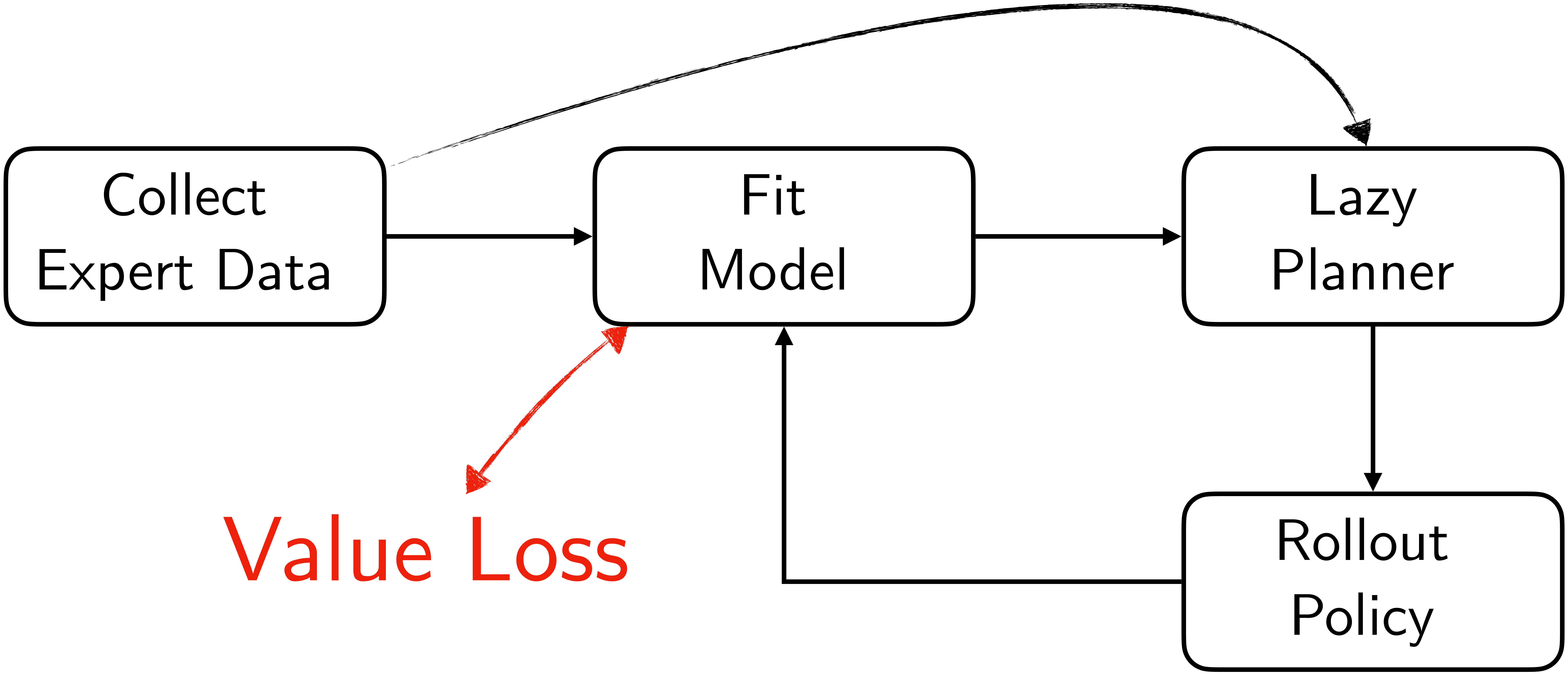
Advantage of expert in model

Value matching on expert states

$$+ T\mathbb{E}_{s,a \sim \hat{\pi}}\left[E_{s' \sim \hat{M}} V^{\hat{\pi}}(s') - E_{s'' \sim M^*} V^{\hat{\pi}}(s'')\right]$$

Value matching on learner states

LAMPS with Moment Matching (LAMPS-MM)



New Lemma: Performance Difference via Advantage in Model

Solution 1:
Be lazy, restart
from expert states

Solution 2:
Match value loss