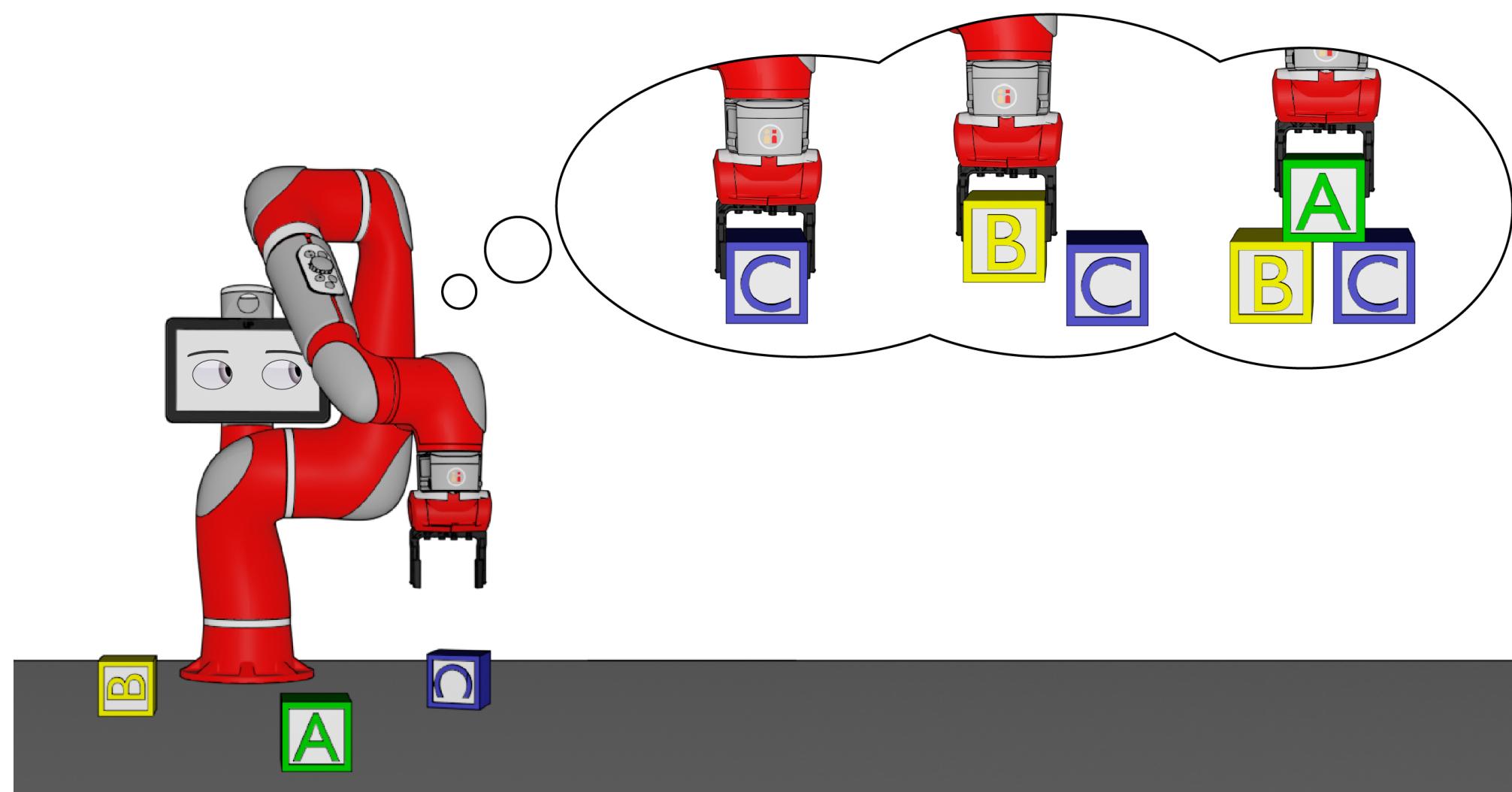


# Reinforcement Learning

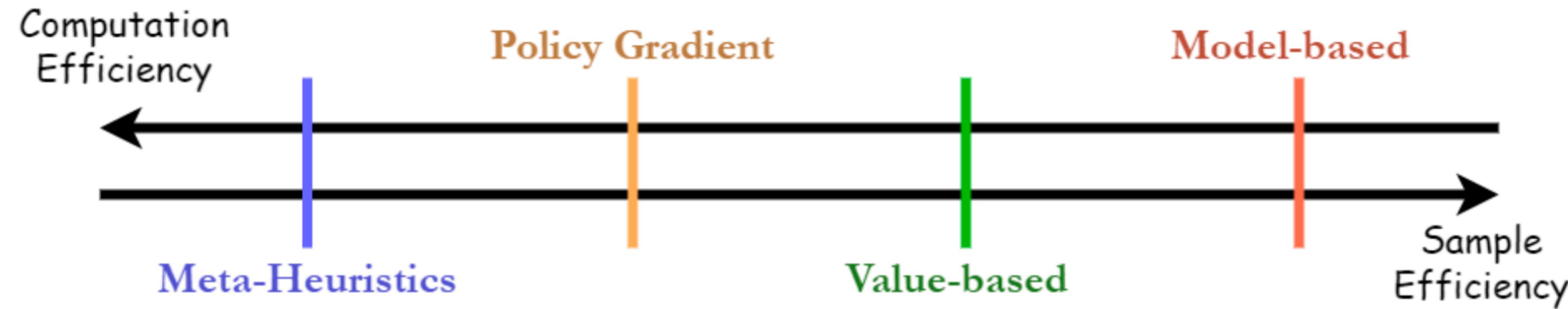
HSE, winter - spring 2025

## Lecture 9: Model-based RL

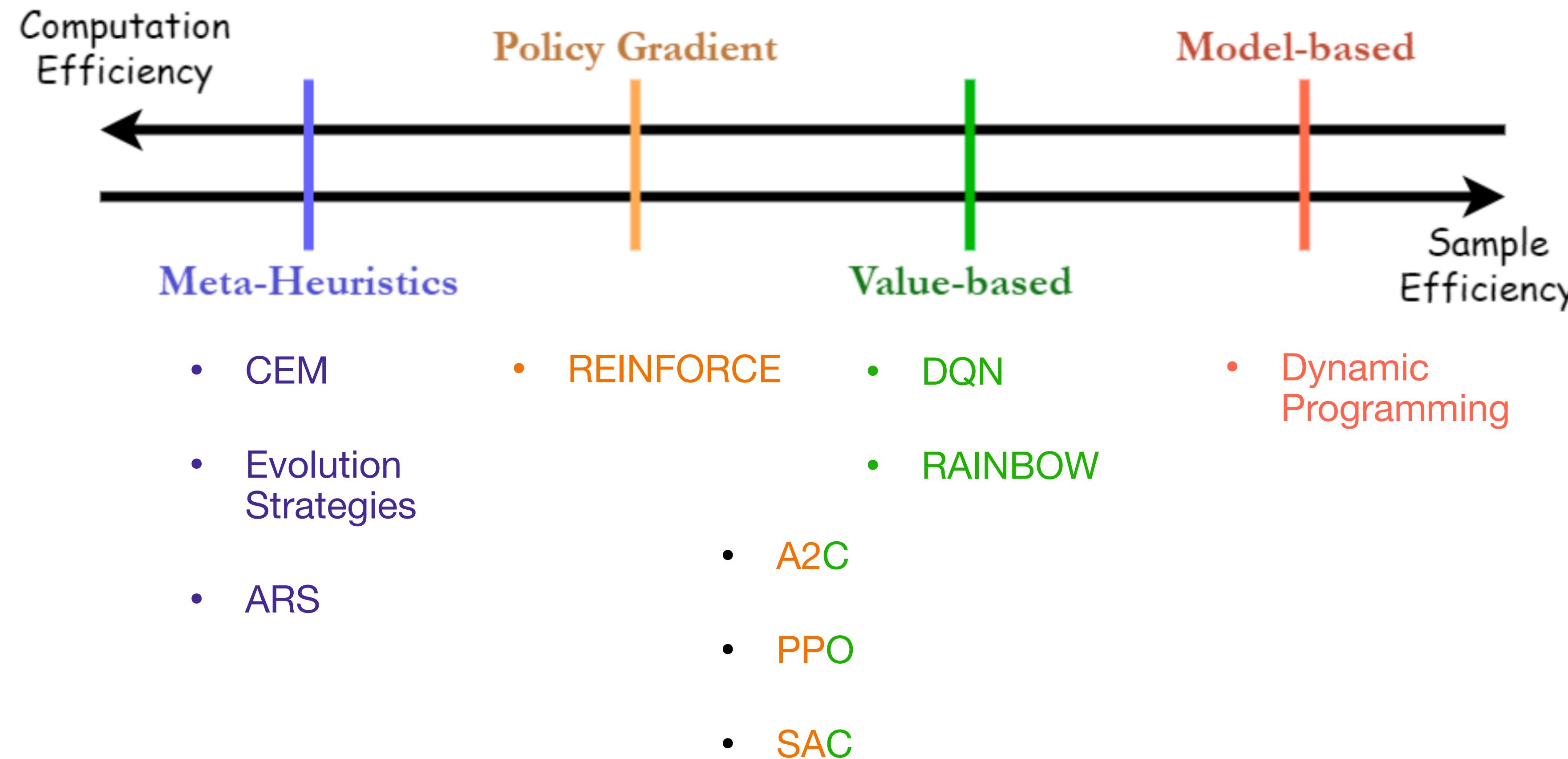


Sergei Laktionov  
[slaktionov@hse.ru](mailto:slaktionov@hse.ru)  
[LinkedIn](#)

# Recap



# Recap



# Model-based Setup

The environment's dynamic is known or learnt:

- Stochastic case:  $p(r, s' | s, a)$
- Deterministic case:  $s' = S(s, a), r = R(s, a)$

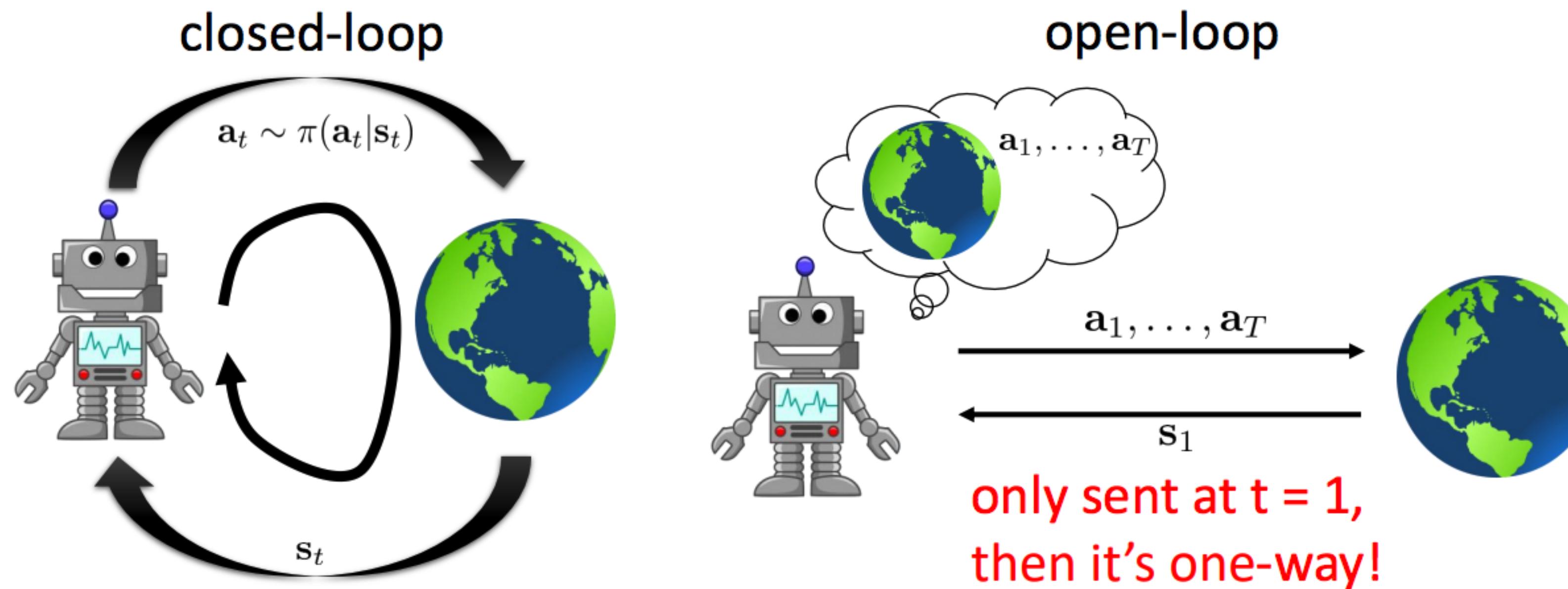
Examples:

- Games (Chess, Go, Shogi etc)

Further in this lecture, we will only consider deterministic environments.

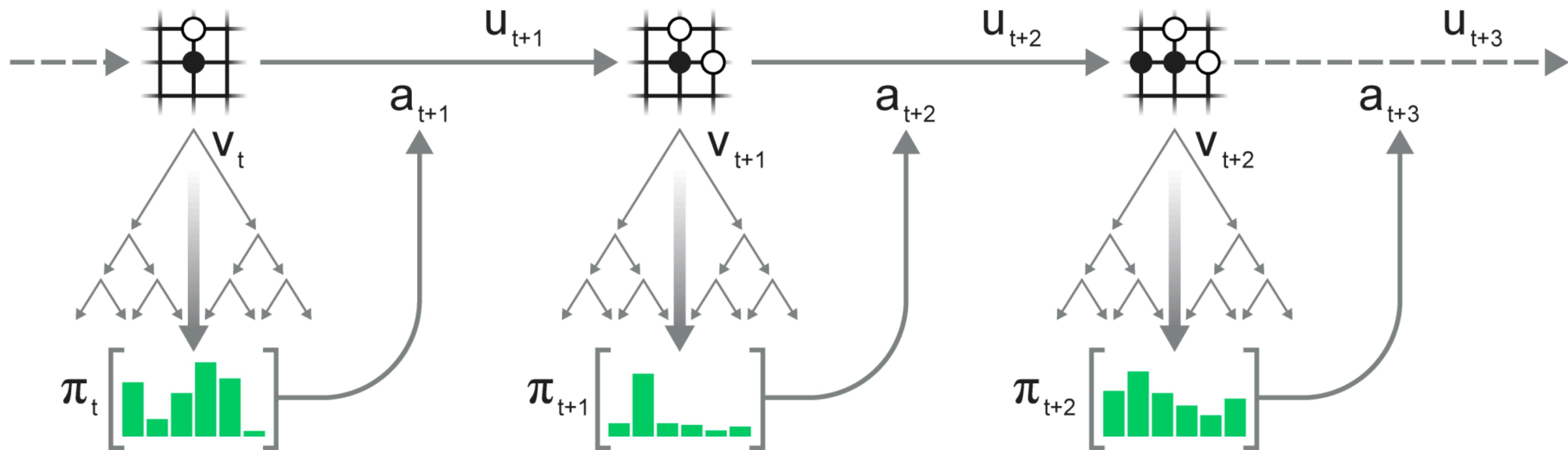
# Planning

Any computational process that takes an environment as input and produces or improves a policy for interacting with the modelled environment:

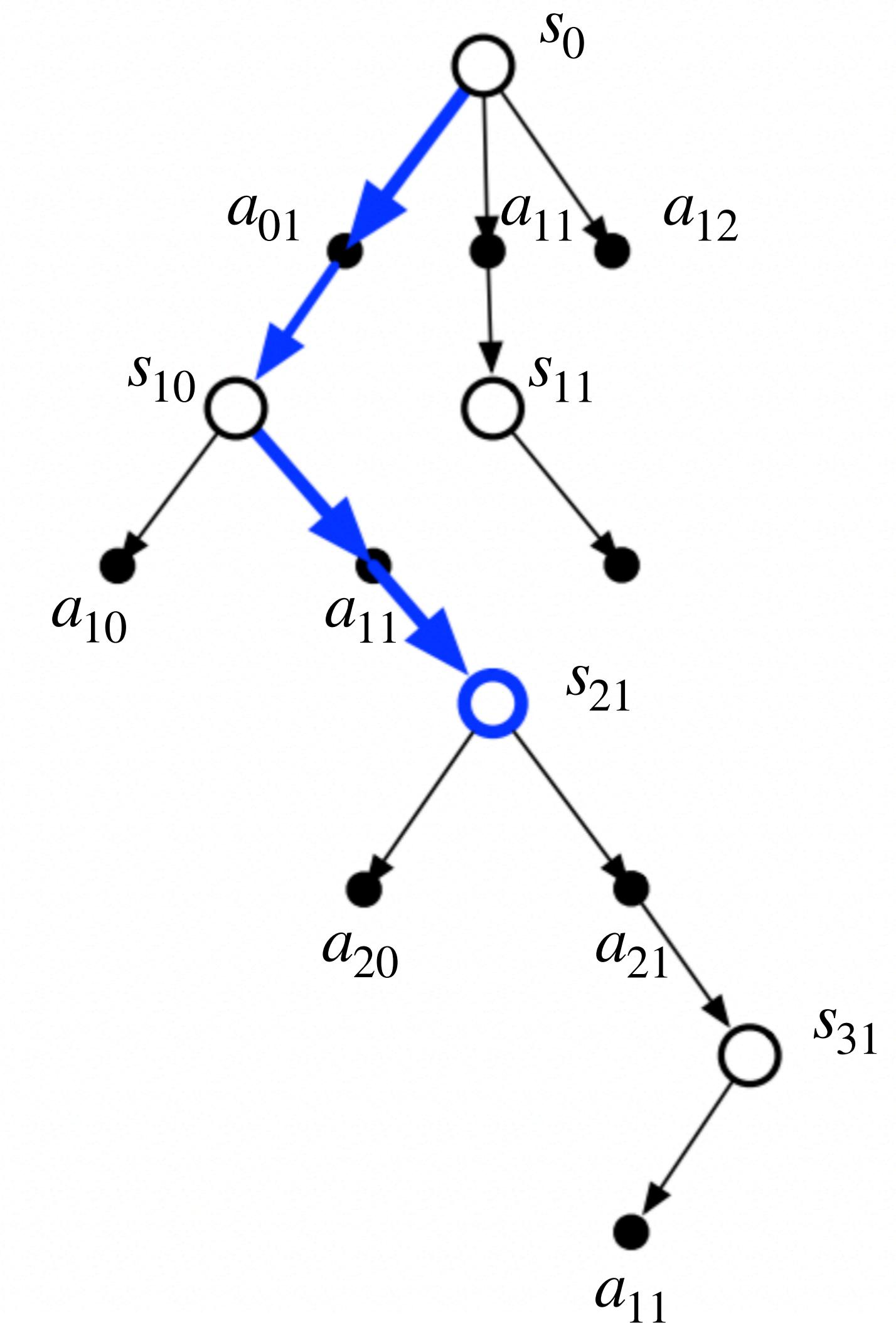


Source

# Planning in the Environment



# Monte-Carlo Tree Search



# Monte-Carlo Tree Search

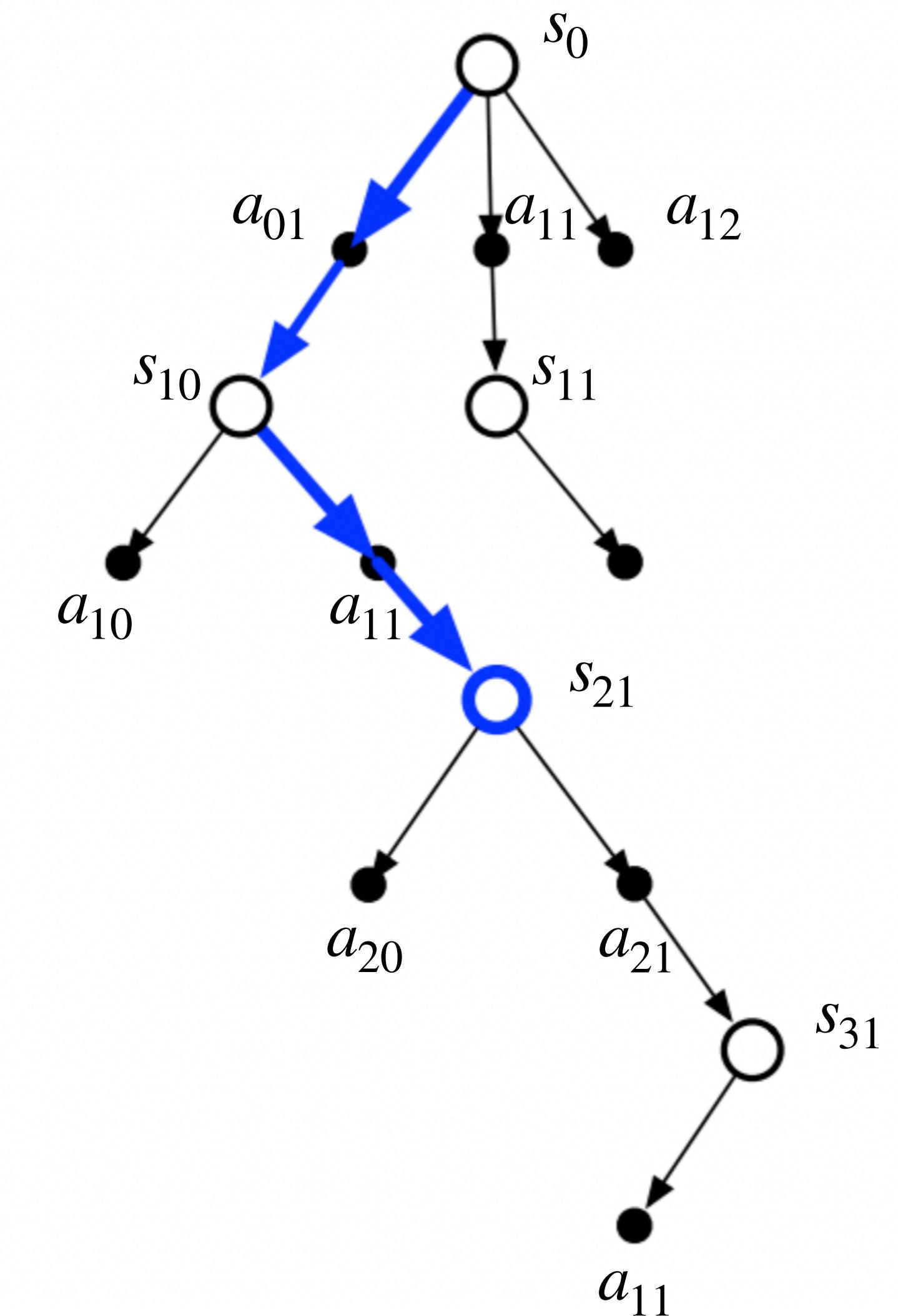
Each node storages a tuple  $(G(s, a), n(s, a))$ .

## Selection:

- Starting from the root, walk down the tree until the leaf node is achieved
- UCB:

$$n(s) = \sum_a n(s, a)$$

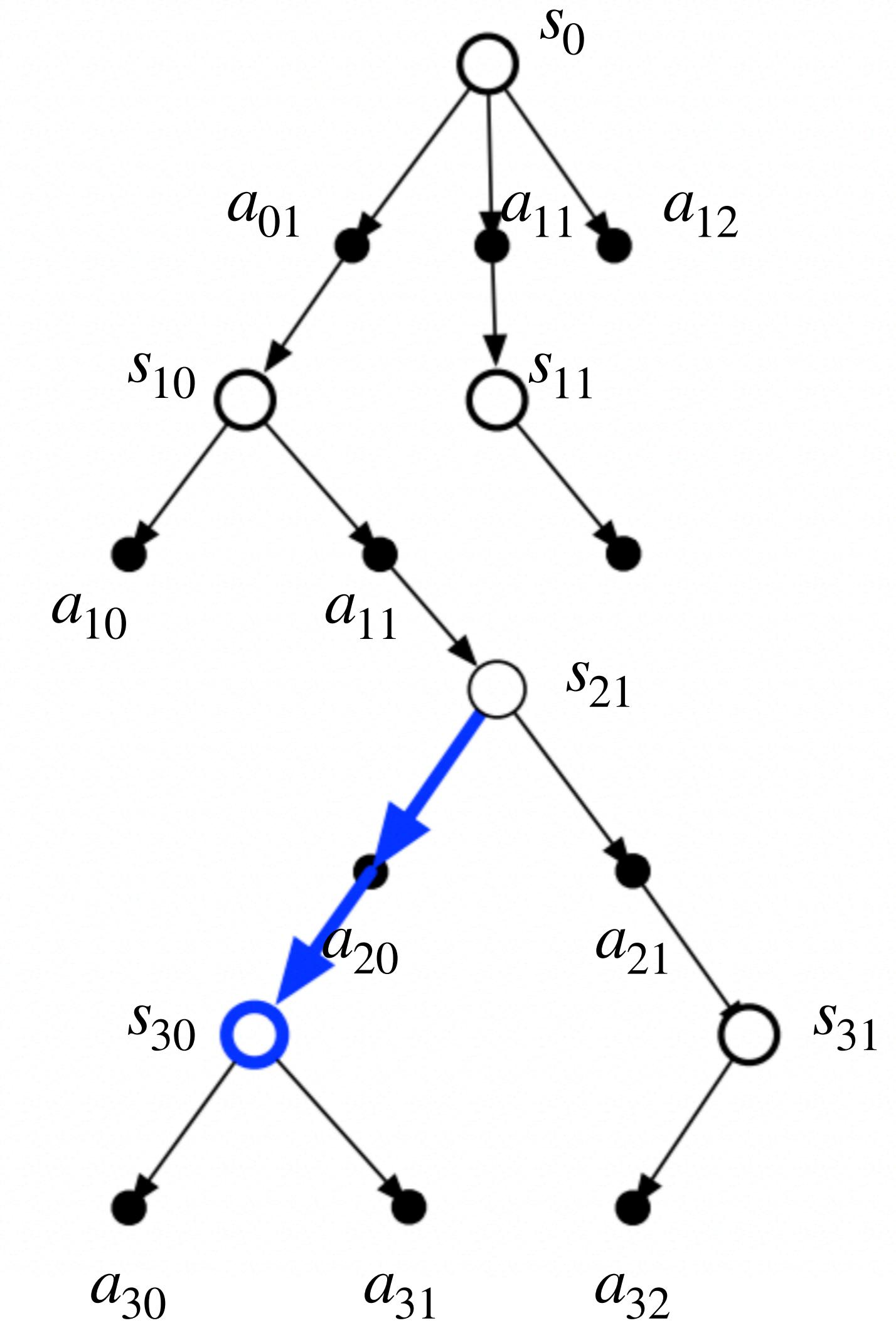
$$\hat{a} = \operatorname{argmax}_a [Q(s, a) + C \sqrt{\frac{\log n(s)}{n(s, a)}}]$$



# Monte-Carlo Tree Search

## Expansion:

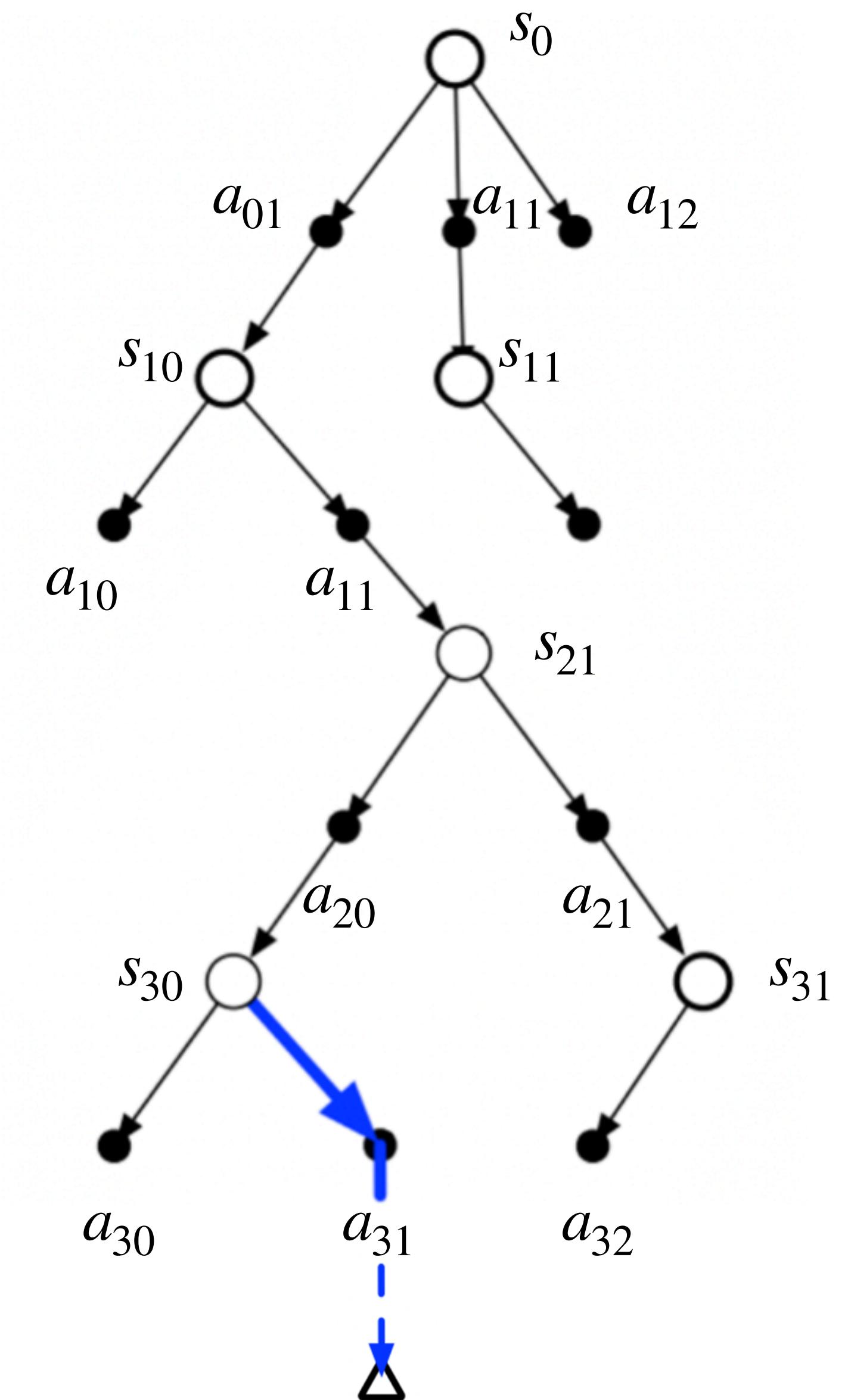
- Select an action  $a$  to apply in the state  $s$ , either randomly or using a heuristic.



# Monte-Carlo Tree Search

## Simulation:

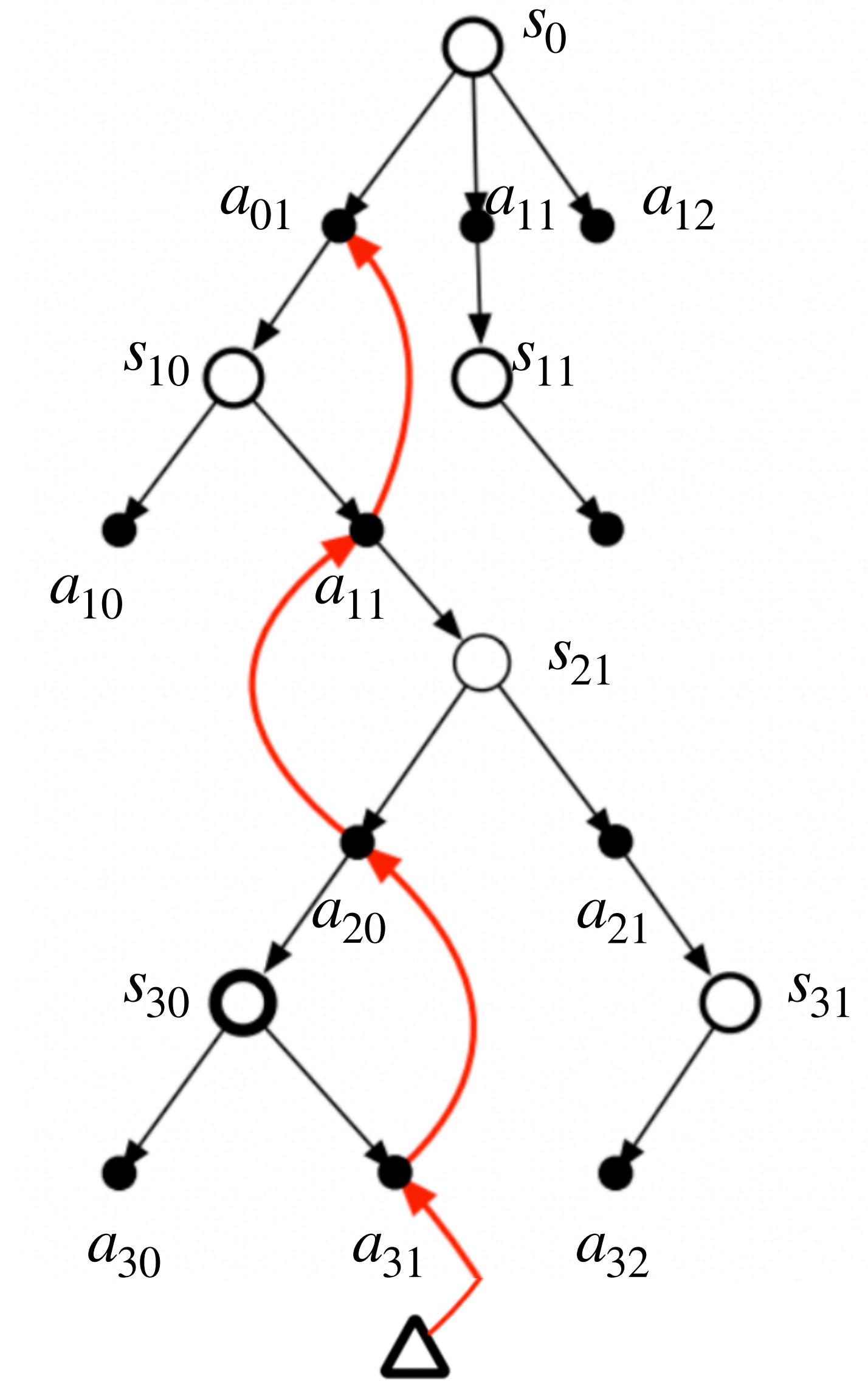
- Perform a randomised simulation of the MDP until we reach a terminating state.



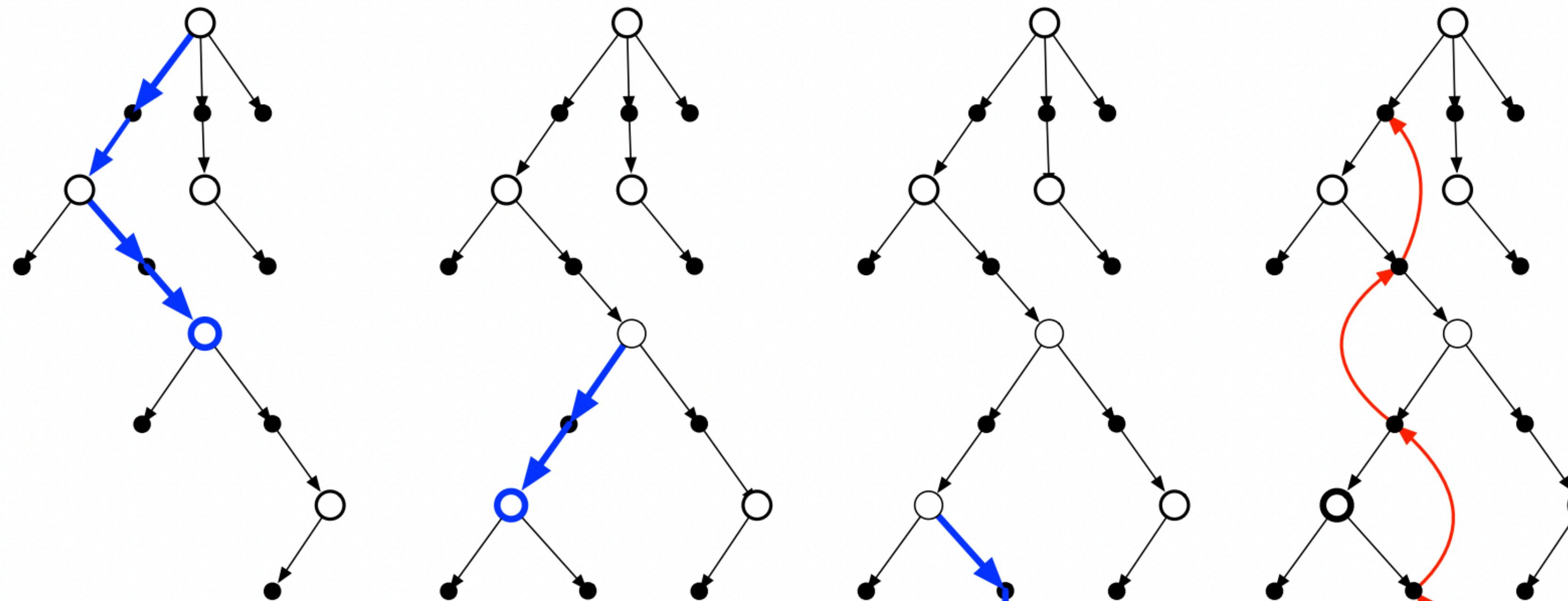
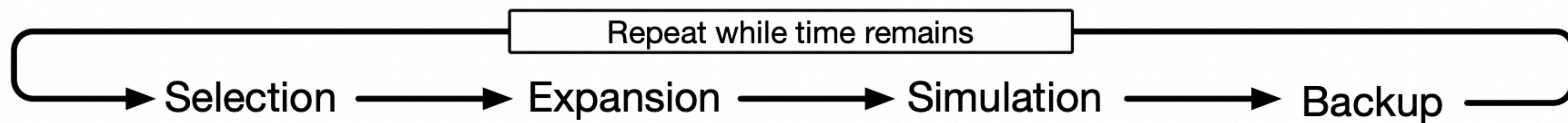
# Monte-Carlo Tree Search

## Propagation:

- $n(s, a) \leftarrow n(s, a) + 1$
- $G \leftarrow r + \gamma G$
- $Q(s, a) \leftarrow Q(s, a) + \frac{1}{n(s, a)}[G - Q(s, a)]$



# Monte-Carlo Tree Search



Tree  
Policy

Rollout  
Policy

# Monte-Carlo Tree Search

The result of the MCTS algorithm is a policy in the current environment's state:

- $\pi_{MCTS}(a | s) \propto n(s, a)^{\frac{1}{\tau}}$
- $\pi_{MCTS}(s) = argmax_a Q(s, a)$

# Self-Play

- The zero-sum game of two players

- $o = \begin{cases} 1, & \text{if now it's first player's turn} \\ -1, & \text{otherwise} \end{cases}$

- $Q_o(s, a) = oQ(s, a)$

- $\hat{a} = argmax_a [Q_o(s, a) + C\sqrt{\frac{\log n(s)}{n(s, a)}}]$

- $\pi_{MCTS}(a | s) \propto n(s, a)^{\frac{1}{\tau}}$

# MCTS

Two issues:

1. After applying an action sample from  $\pi_{MCTS}(\cdot | s)$  we probably remove the tree's nodes corresponding to not chosen actions.
2. We must play till the end to get an estimation of  $V(s)$ .

# MCTS

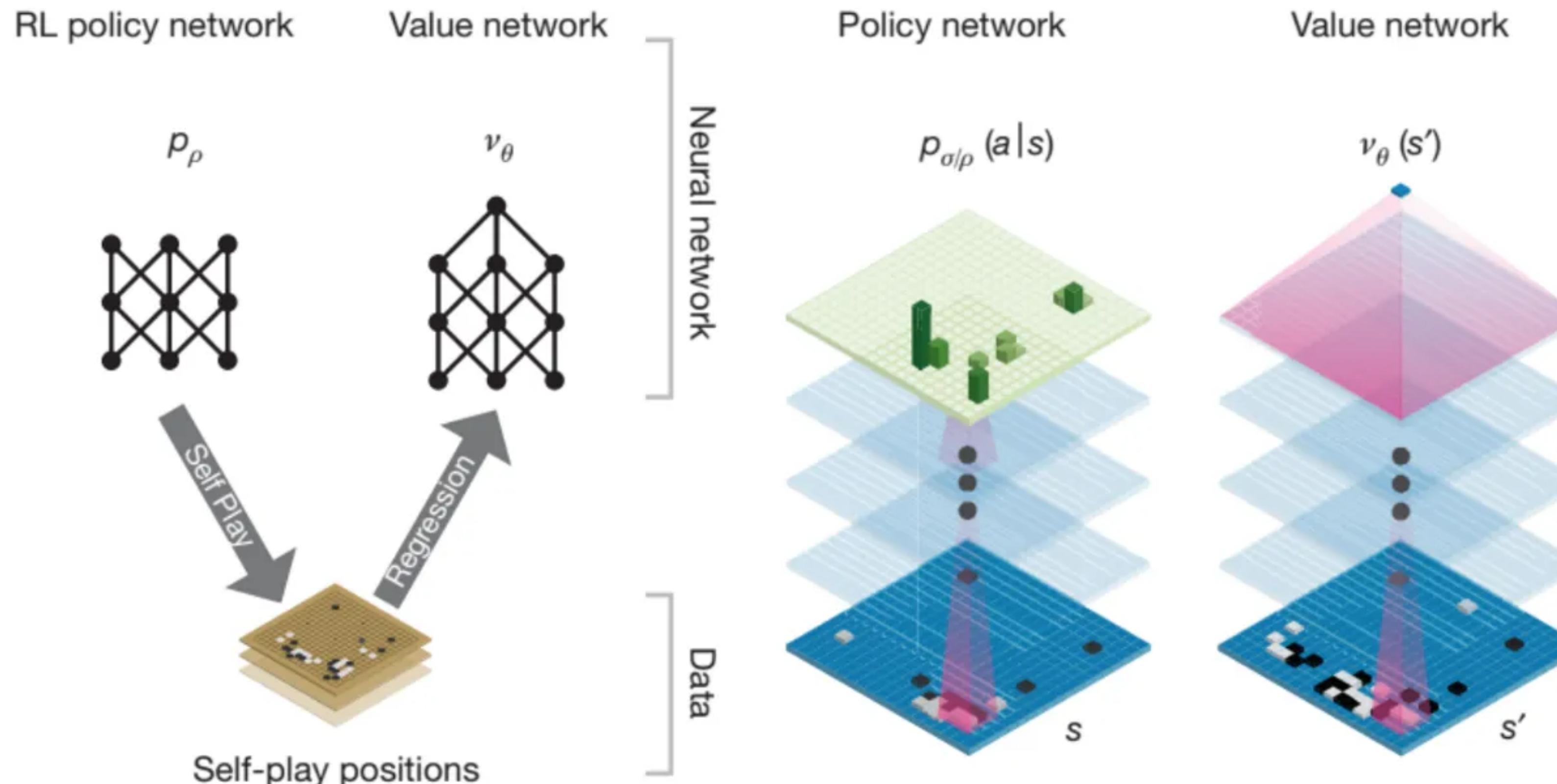
Two issues:

1. After applying an action sample from  $\pi_{MCTS}(\cdot | s)$  we probably remove the tree's nodes corresponding to not chosen actions.
2. We must play till the end to get an estimation of  $V(s)$ .

Two ideas:

1. Distill  $\pi_{MCTS}$  in  $p_\theta$ .
2.  $V \approx V_\phi$

# Architecture

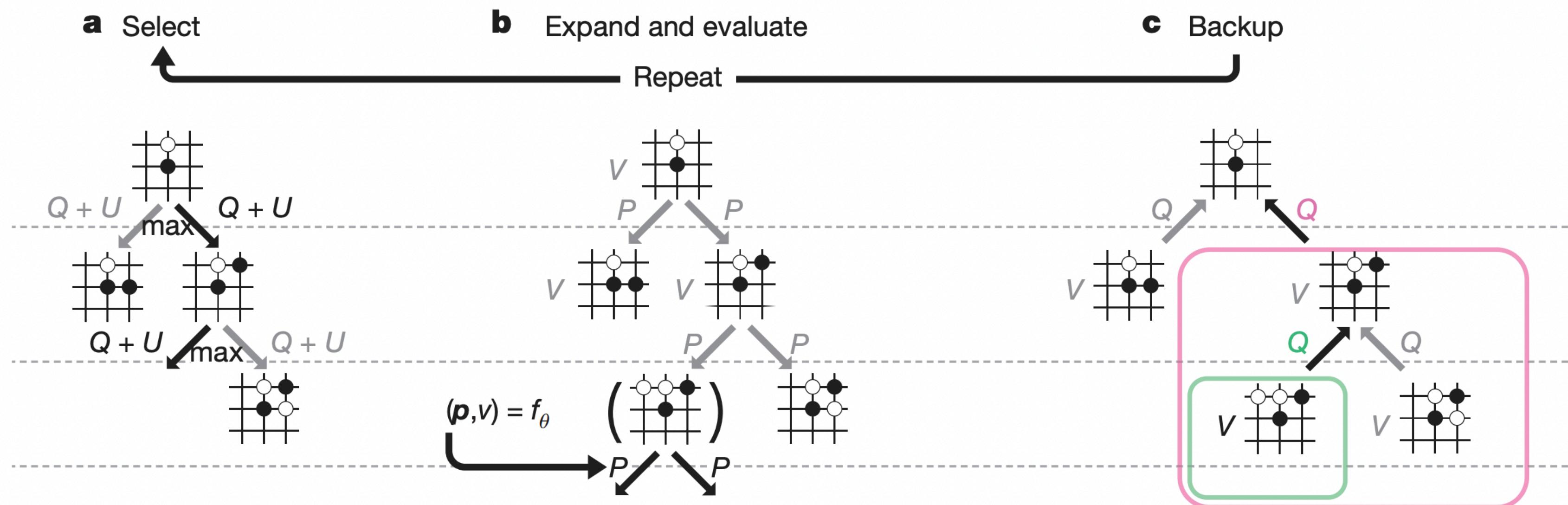


Policy and Value networks are indeed combined

# MCTS in AlphaZero

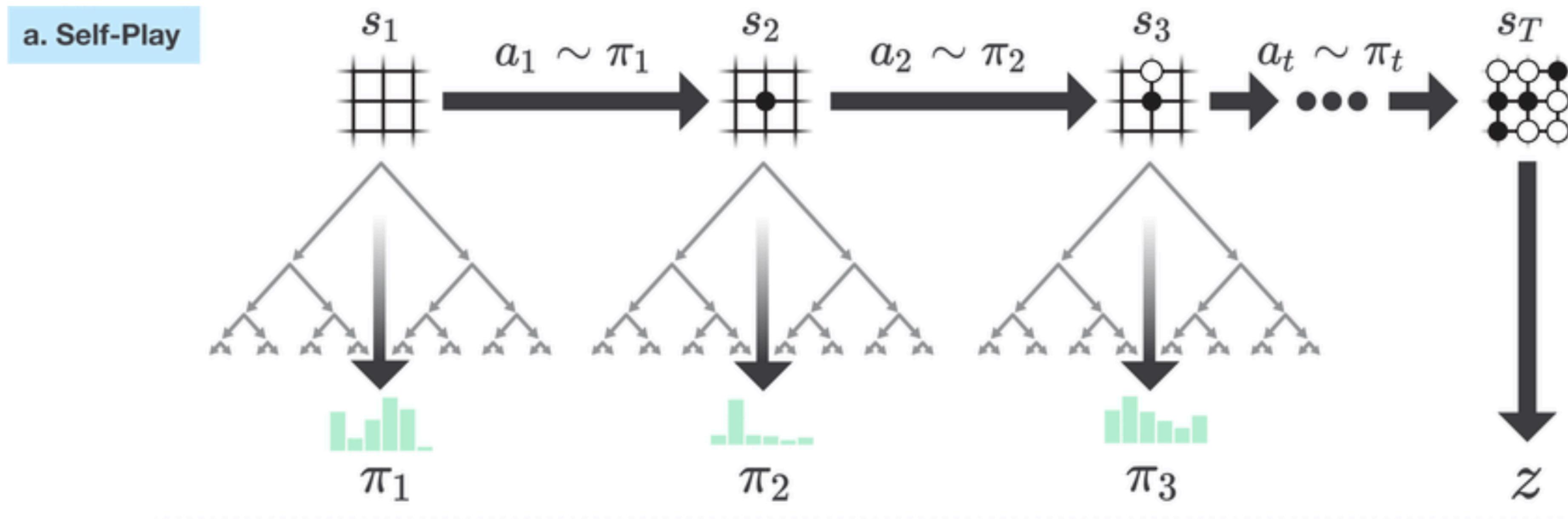
$f_\theta(s) = (\mathbf{p}, V)$  is a two-head neural network.

1. Selection:  $\hat{a} = \operatorname{argmax}_a [\hat{Q}(s, a) + C p_\theta(a | s) \frac{\sqrt{n(s)}}{n(s, a) + 1}]$
2. Expansion
3. Propagation: for leaf state  $s_{leaf}$  use  $V_\theta(s_{leaf})$



# AlphaZero Network Training

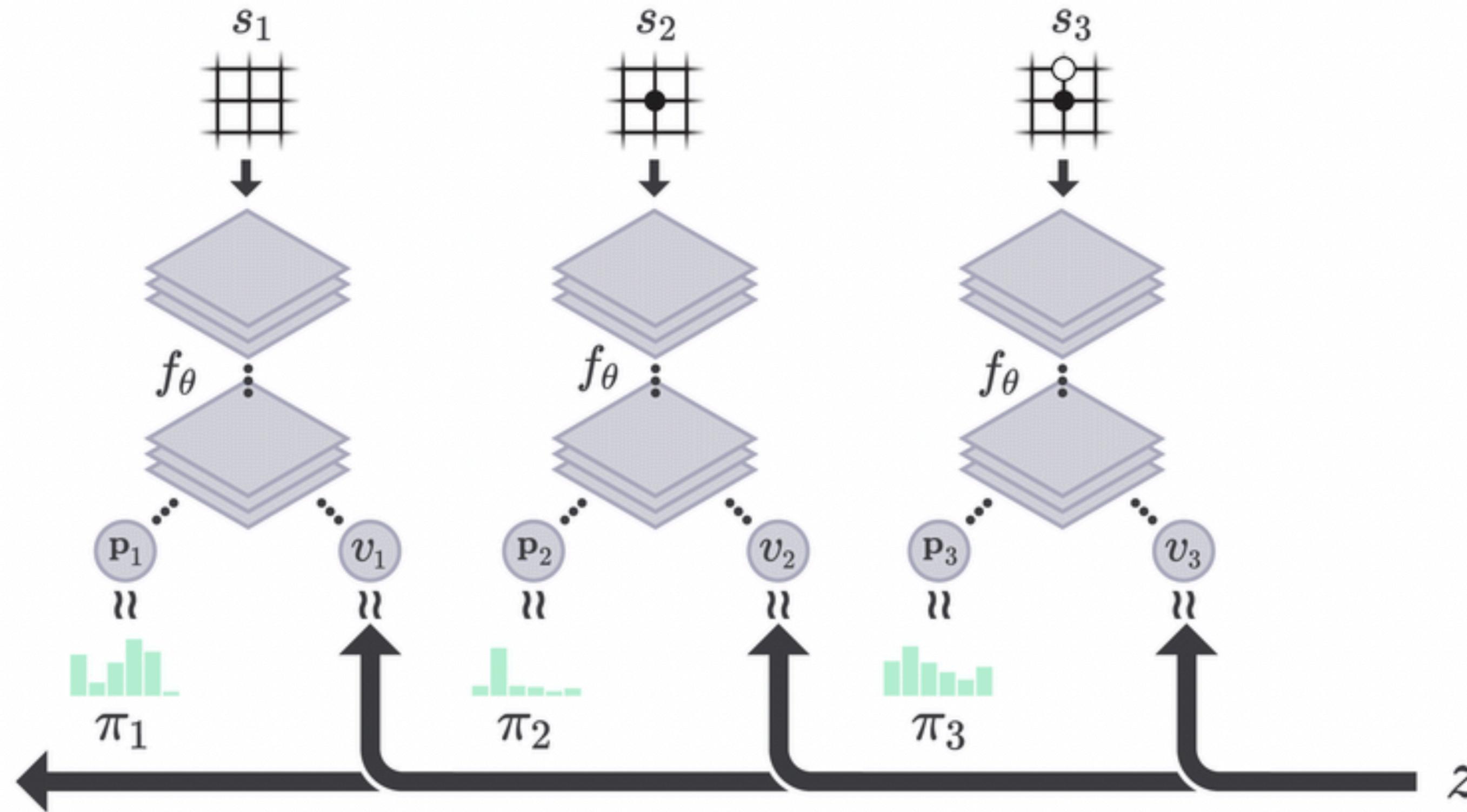
1. Many self-play games: in each state, an agent plans with MCTS and executes action  $a \sim \pi(\cdot | s) \propto n(s, a)^{\frac{1}{\tau}}$
2. Store the dataset of triplets  $(s, \pi_{MCTS}(\cdot | s), z)$ .



# AlphaZero Network Training

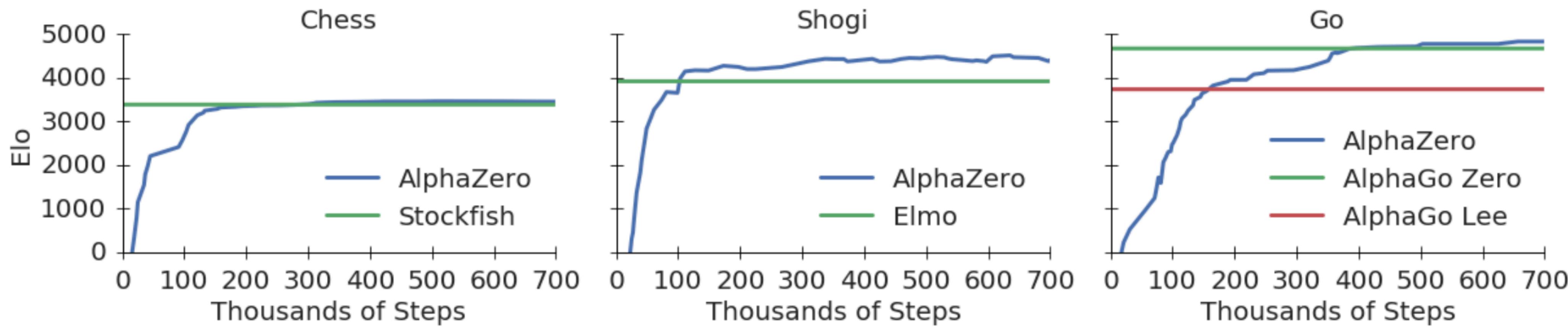
b.

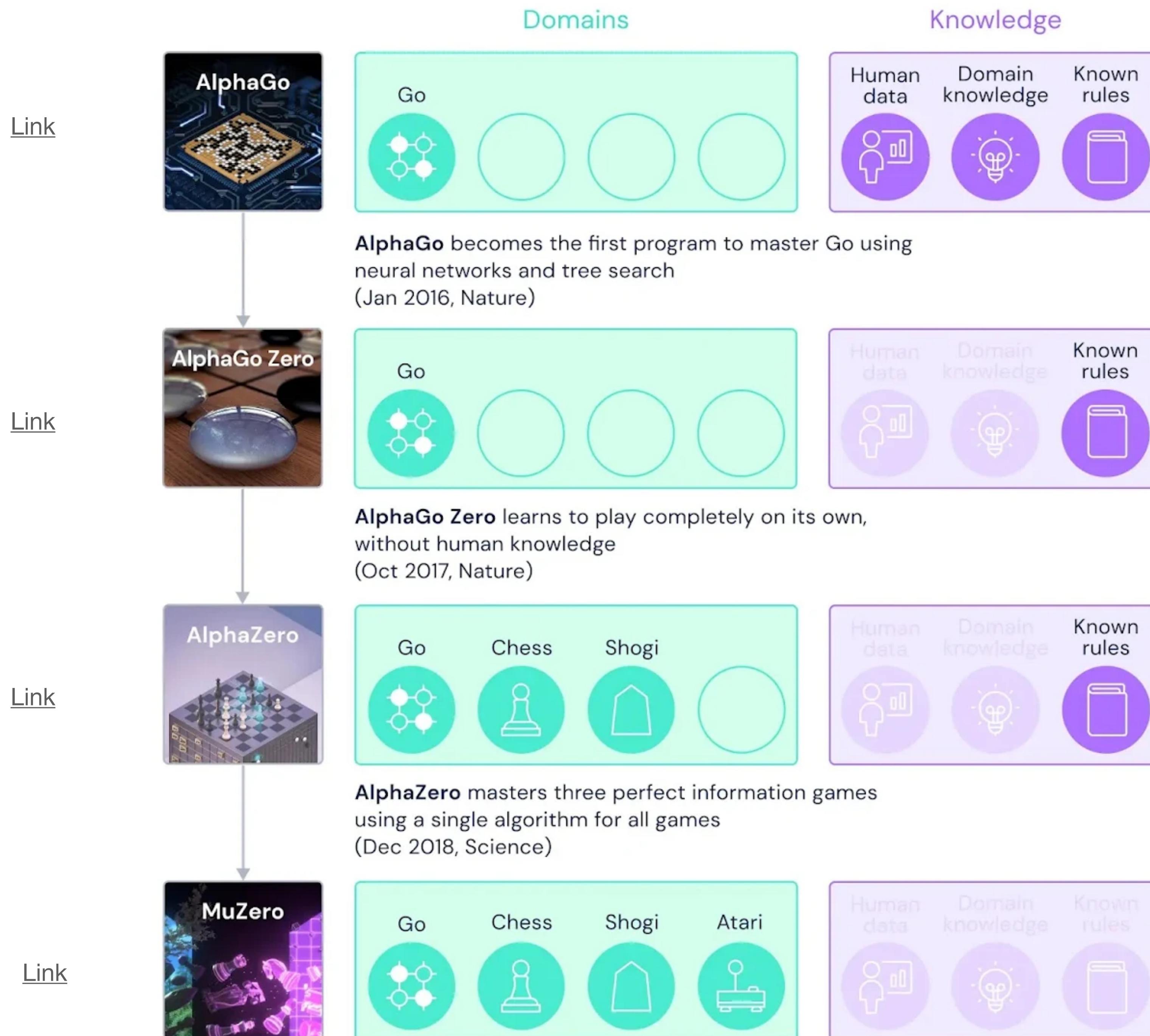
Neural Network Training



$$(z - V_\theta(s))^2 + \sum_a \pi_{MCTS}(s | a) \log p_\theta(a | s) + c ||\theta||_2^2 \rightarrow \min_{\theta}$$

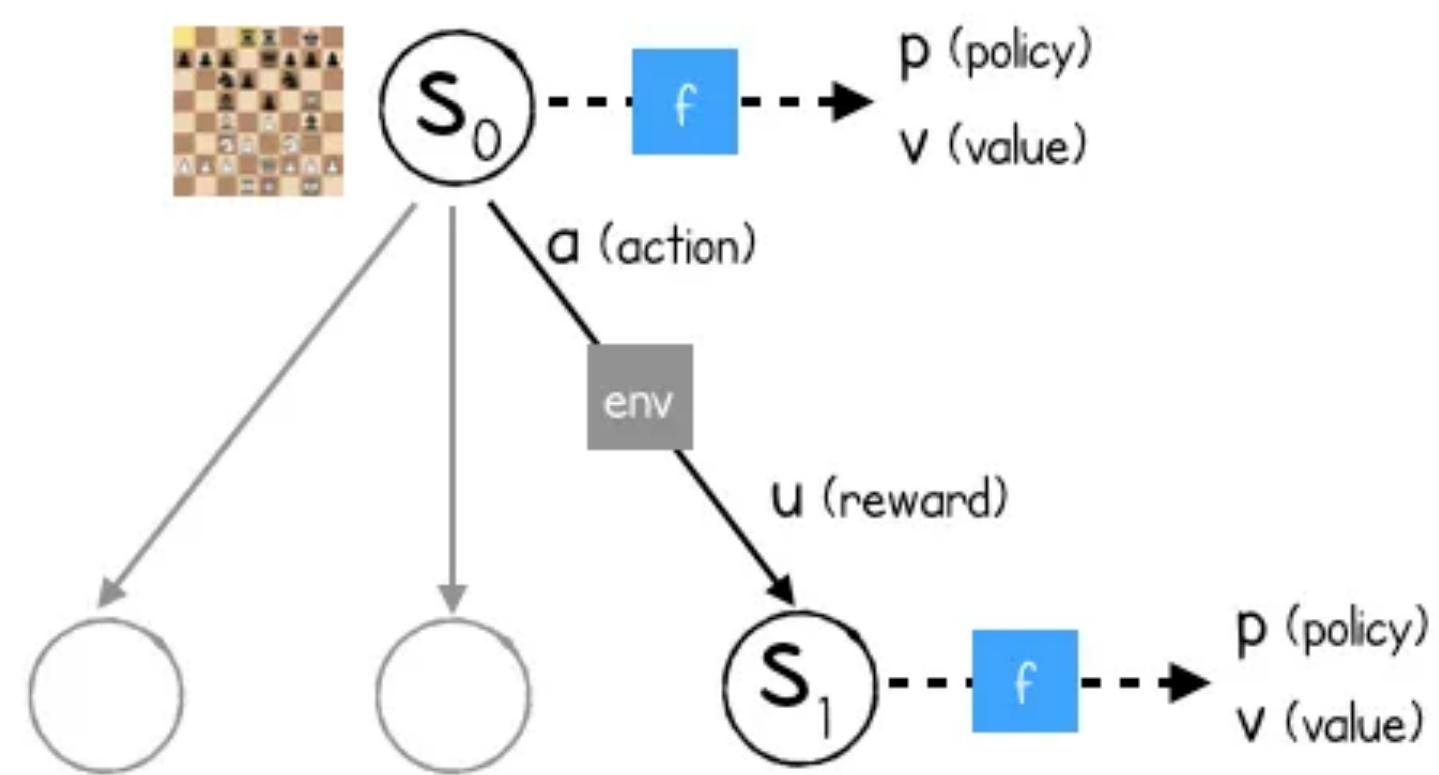
# AlphaZero Results





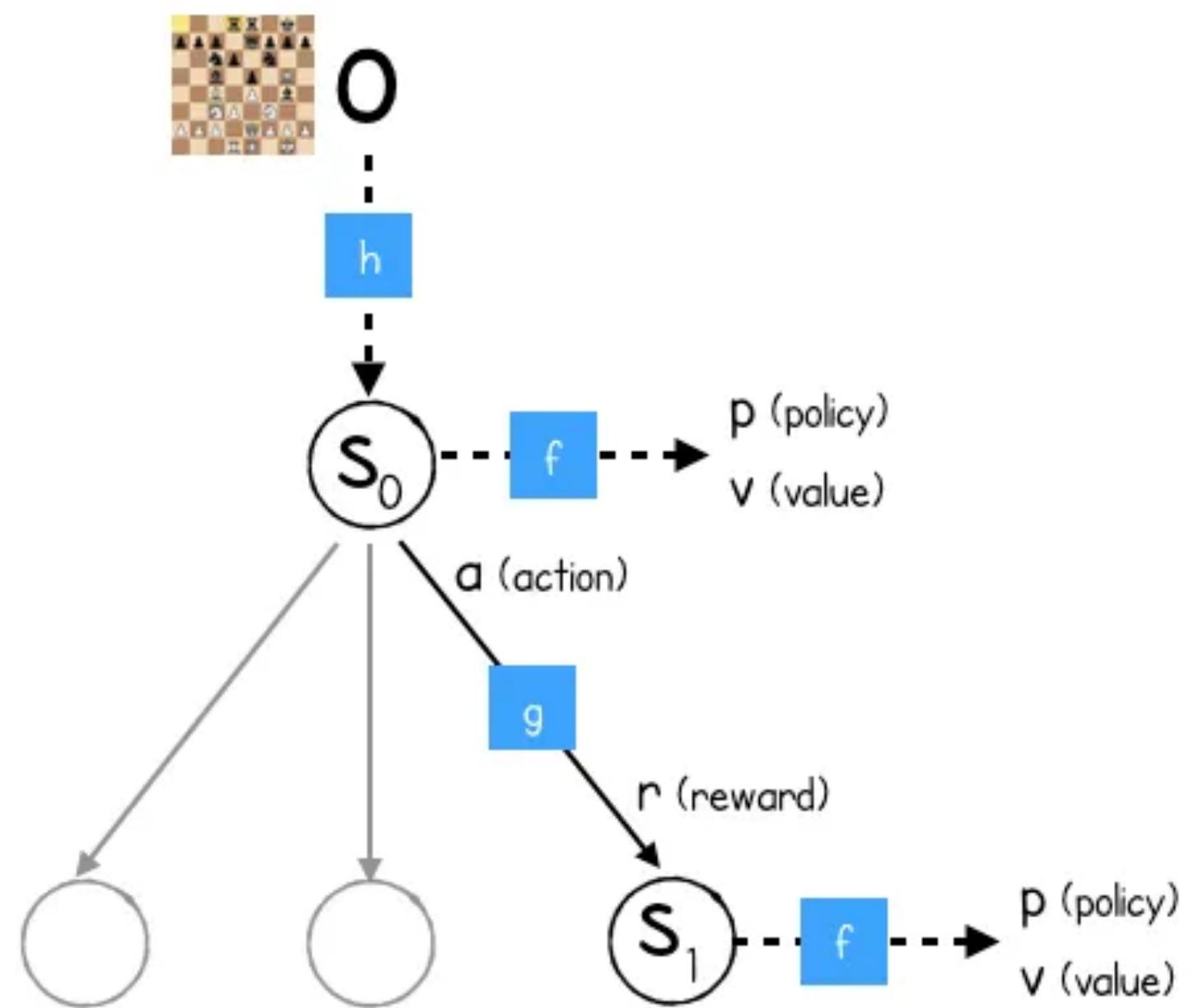
# Comparison

## AlphaZero



AlphaZero has 1 network  
**prediction**  $f$ : from  $s$  to  $p, v$

## MuZero



MuZero has 3 networks  
**prediction**  $f$ : from  $s$  to  $p, v$   
**dynamics**  $g$ : from  $s, a$  to  $r, s$   
**representation**  $h$ : from  $o$  to  $s$

- Representation:

$$h_{\theta}(o_1, \dots, o_t) = s_t^0$$

- Prediction:

$$f_{\theta}(s_t^k) = (\mathbf{p}_t^k, v_t^k)$$

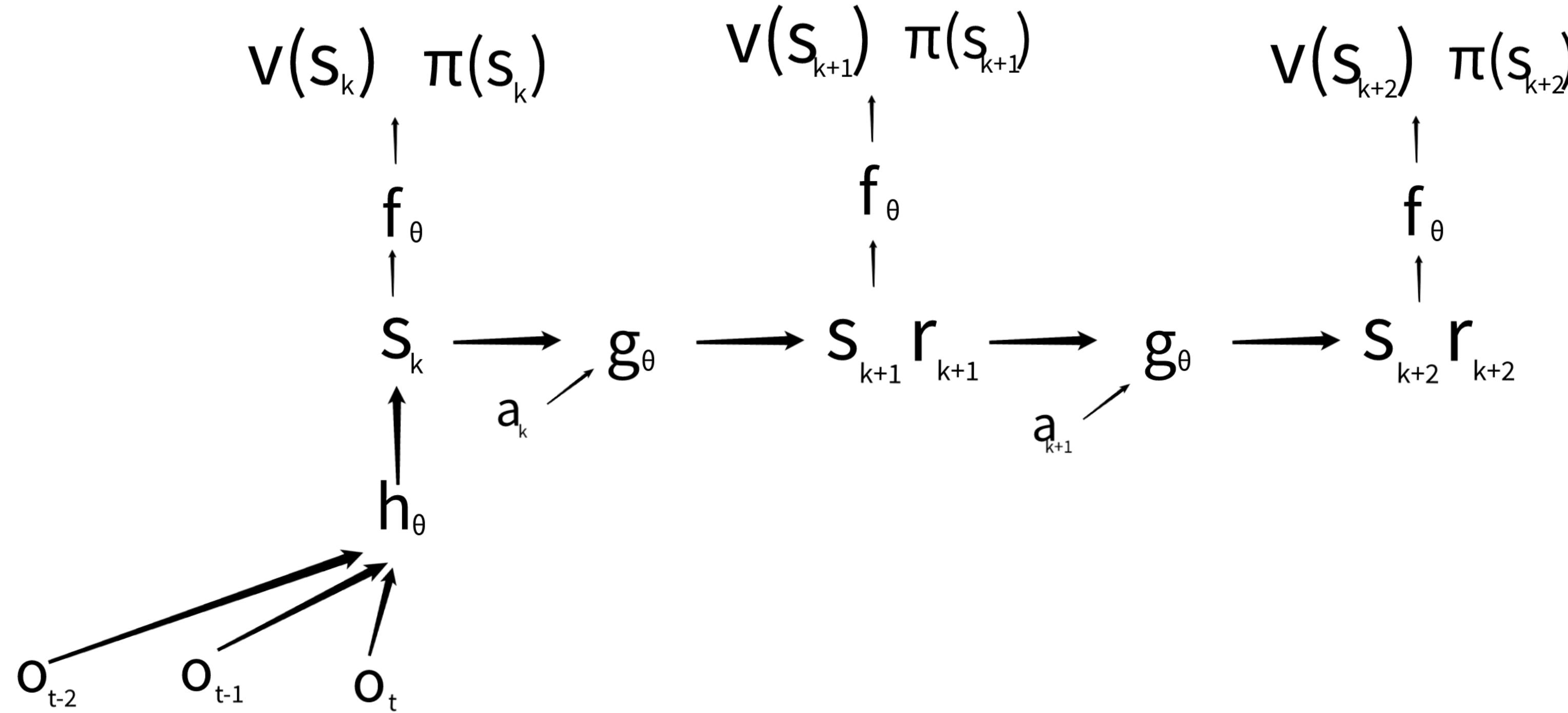
- Dynamics:

$$g_{\theta}(s_t^{k-1}, a_t^k) = (r_t^k, s_t^k)$$

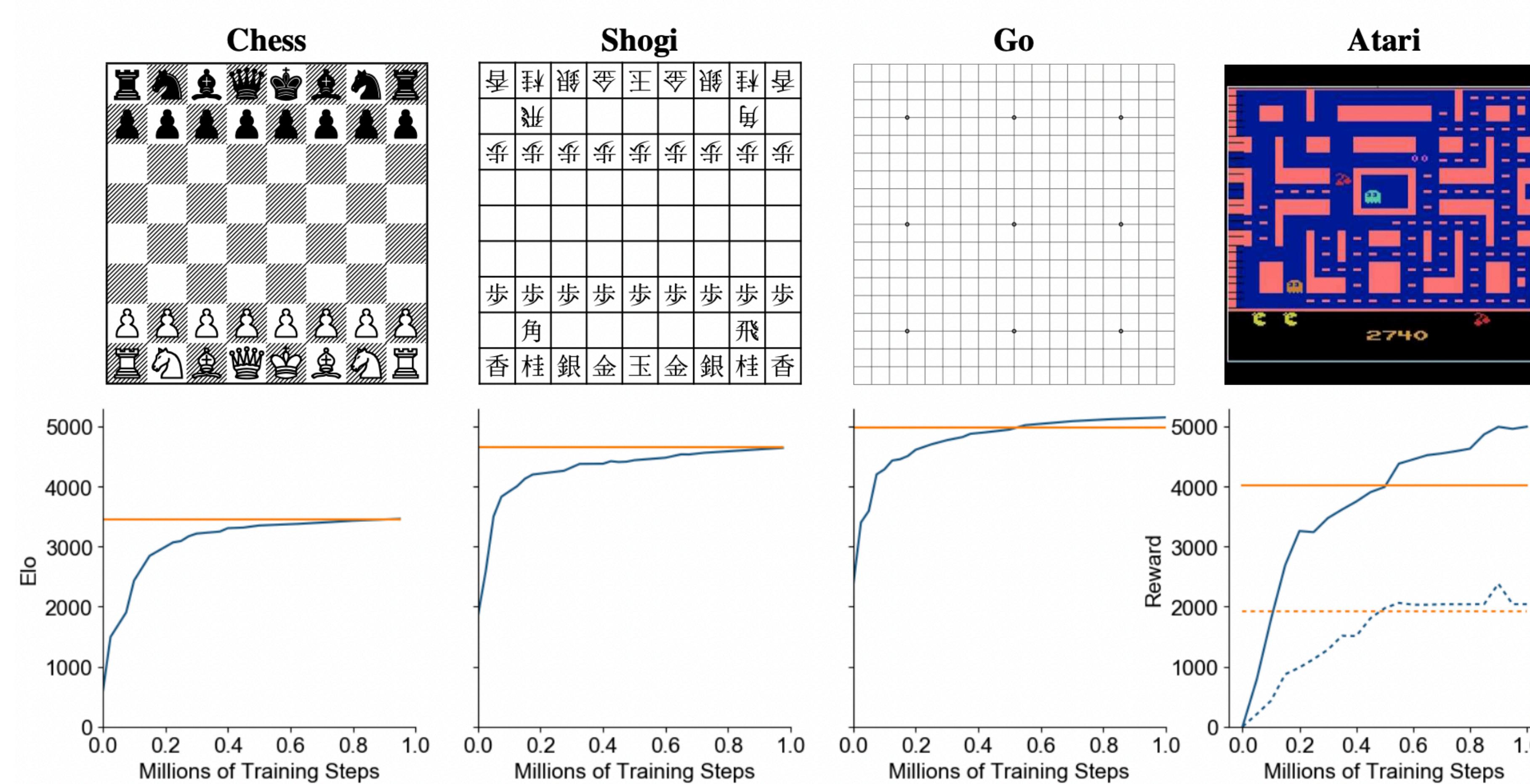
# MuZero Loss

$$l_t(\theta) = \sum_{k=0}^K [(u_{t+k} - r_t^k)^2 + (z_{t+k} - v_t^k)^2 + \pi_{t+k} \log p_t^k] + c \|\theta\|_2^2 \rightarrow \min_{\theta}$$

- $u_{t+k}$  is an immediate reward
- $z_{t+k}$  is  $n$ -step Bellman target



# MuZero Results



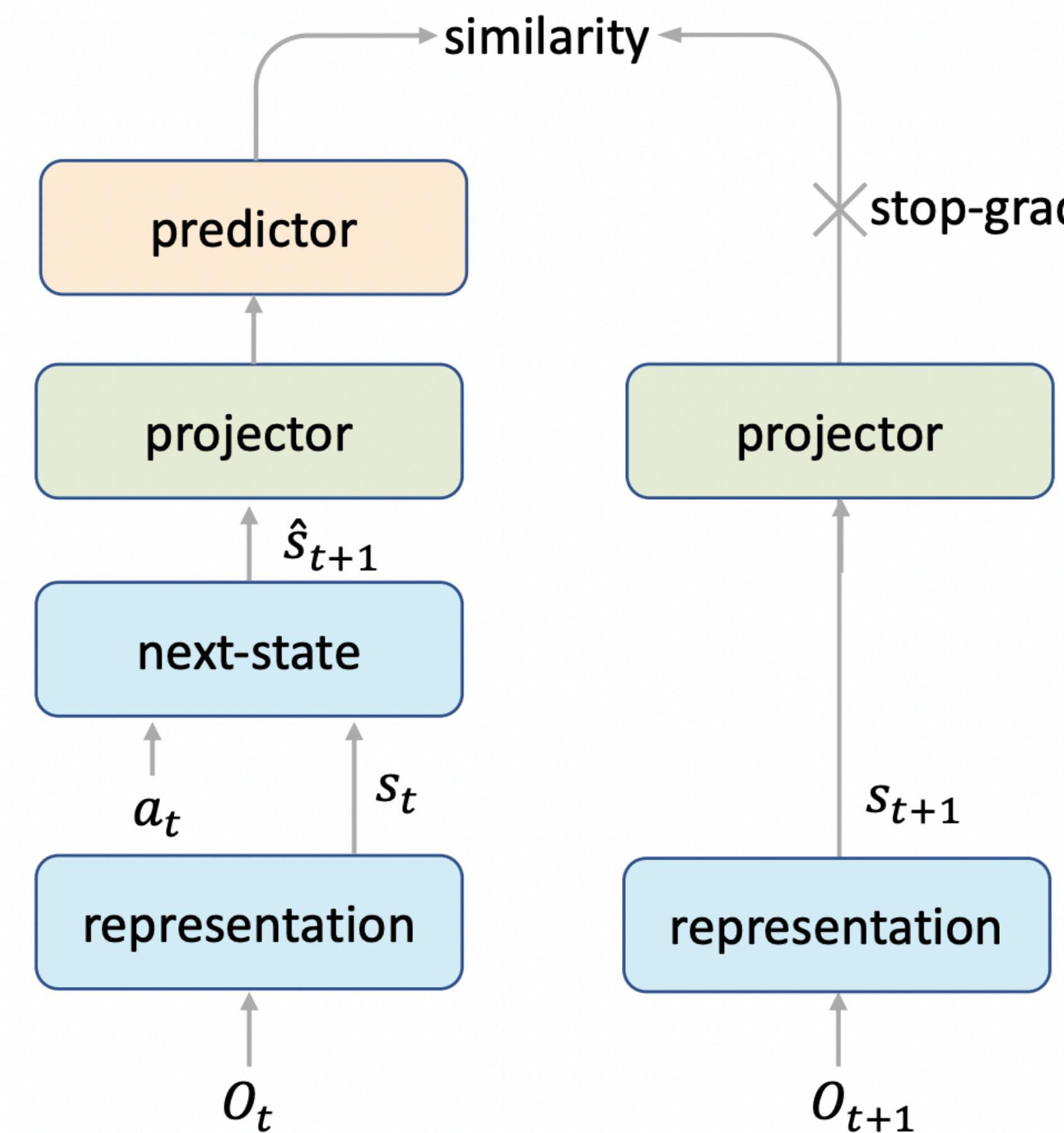
# EfficientZero

MuZero issues:

1. Lack of supervision on environment model:
  - The reward can be sparse
  - Targets are calculated with bootstrapping
2. Hardness to deal with aleatoric uncertainty:
  - the predicted rewards still have large prediction errors.
3. Off-policy issues of multi-step value

# Self-Supervised Consistency Loss

1. Compare the latent representation of consecutive observations with negative cosine similarity and optimise for it.



# End-To-End Prediction of the Value Prefix

2. Predict the value prefix, not immediate rewards:

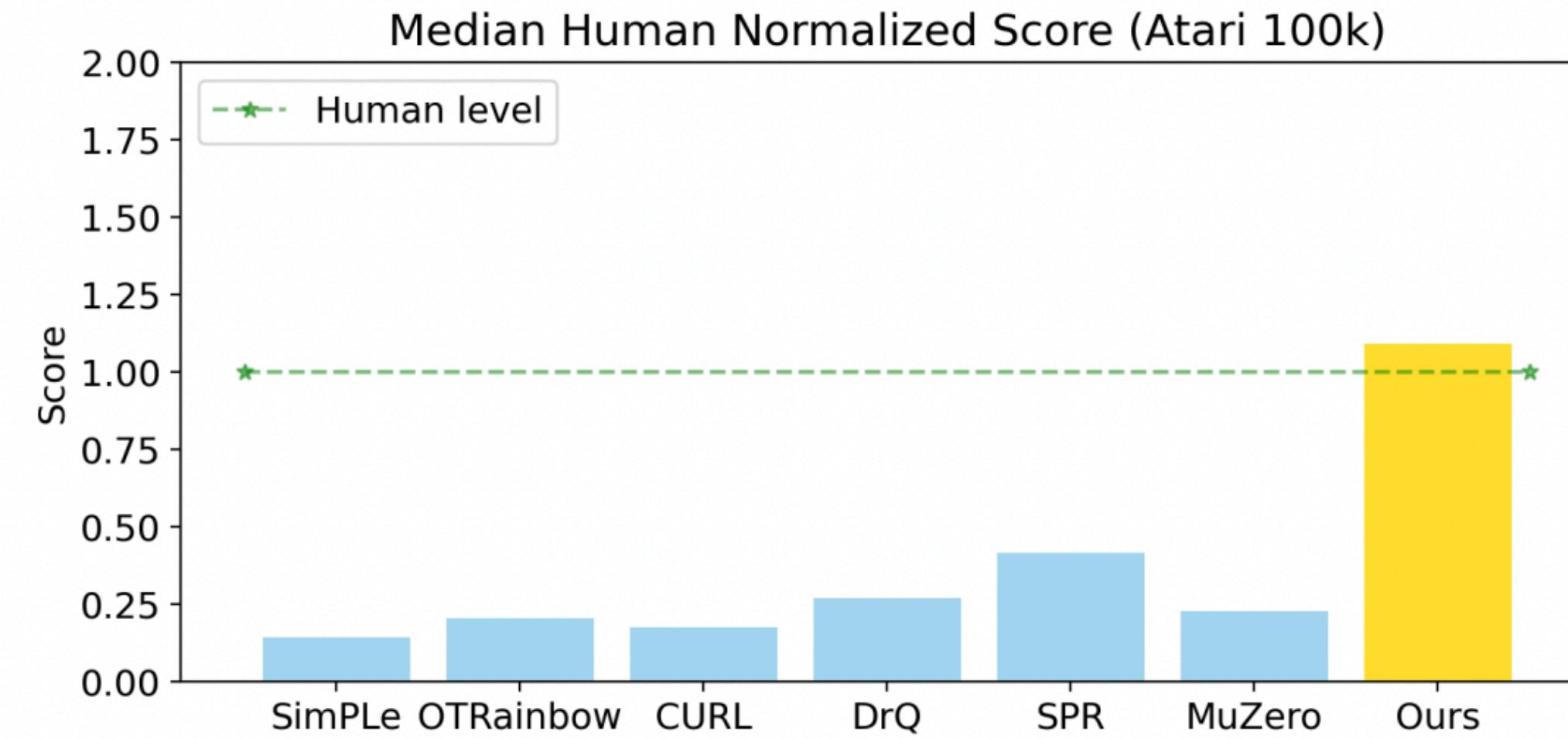
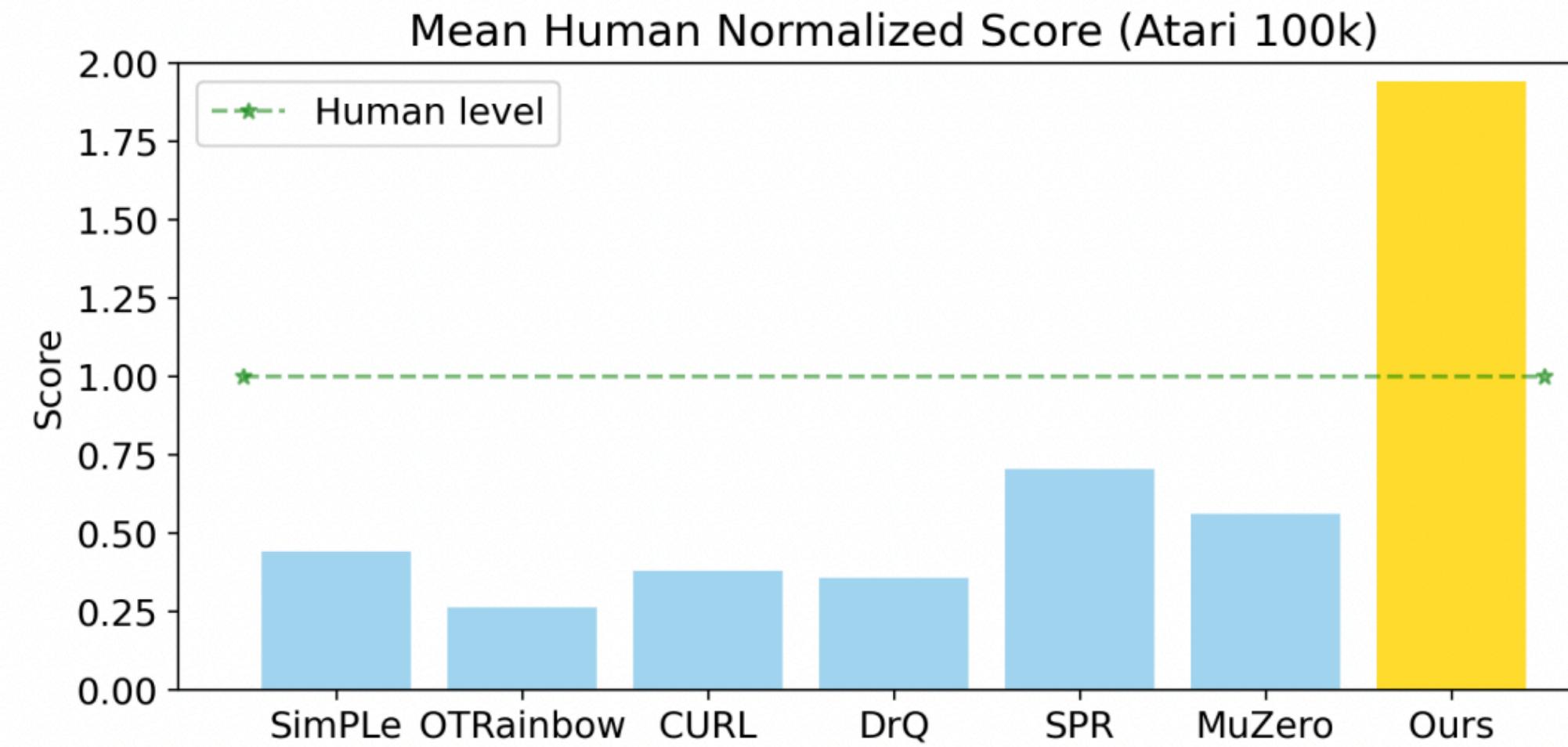
$$Q(s_t, a) = \boxed{\sum_{k=0}^{K-1} \gamma^k r_{t+k}} + \gamma^K V(s_{t+K})$$

# Model-Based Off-Policy Correction

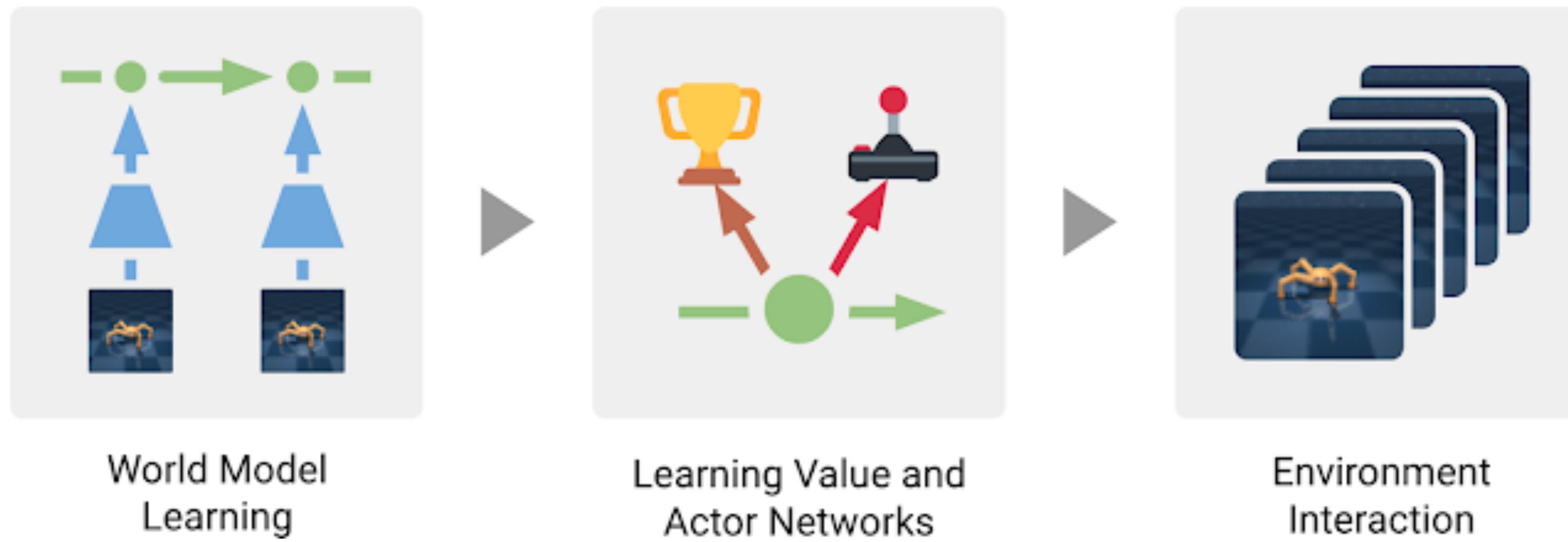
3. Use rewards of a dynamic horizon  $L$  from the old trajectory and redo an MCTS search with the current policy on the last state  $s_{t+L}$  and compute the empirical mean value at the root node.:

$$z_t = \sum_{k=0}^{L-1} \gamma^k u_{t+k} + \gamma^L V^{MCTS}(s_{t+L}), L < K$$

# EfficientZero Results



# Dreamer-V1



# Learning the World Model



# Efficient Behaviour Learning



$o_1$

# Background

1. Practical RL course by YSDA, week 10
2. Reinforcement Learning Textbook (in Russian): 7.2-7.3
3. DreamerV3
4. EfficientZeroV2

**Thank you for your attention!**