RL 2.0

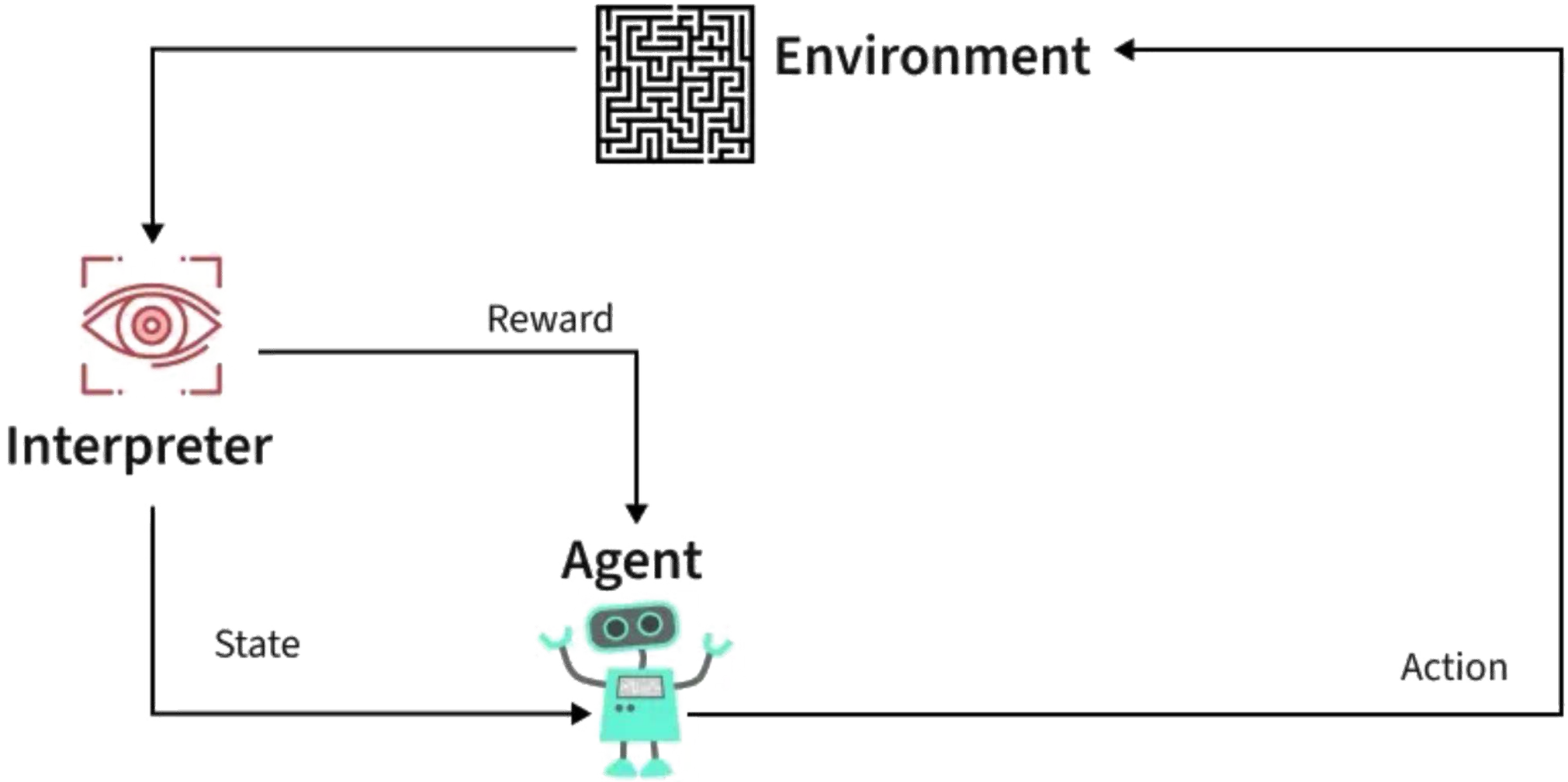
by Ethan Kolasky

quote 01

> Reinforcement learning is the discipline of developing AI systems that can improve based on their performance in realistic environments.

What is Reinforcement Learning?

fig 01



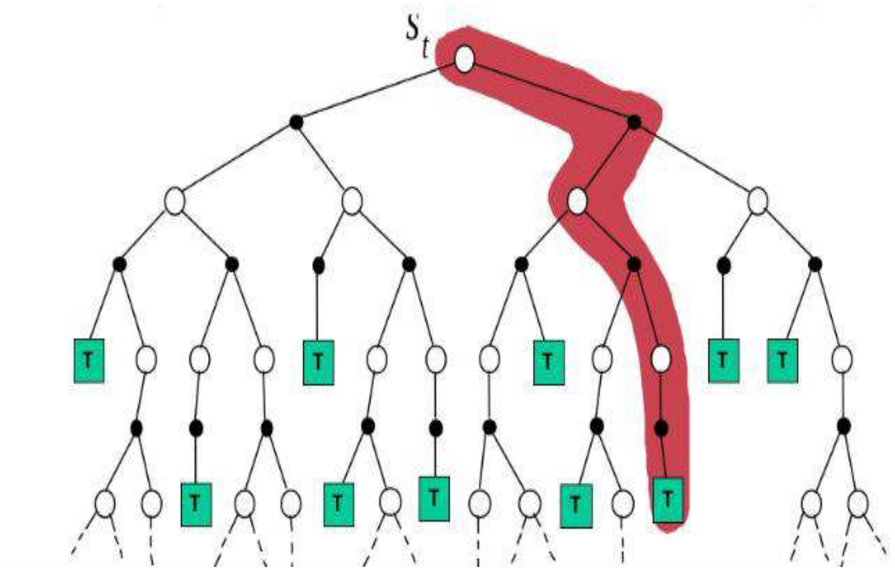
Definitions

Key terms and ideas in reinforcement learning.

Model-Based versus Model-Free

Model-Based

Using a model to exactly determine the optimal next move.

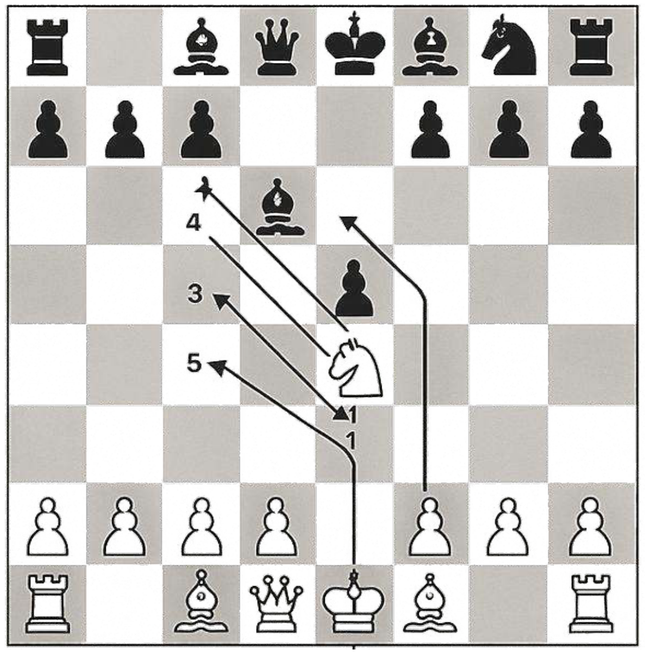


Model-Free

Planning without direct access to a model of the environment.

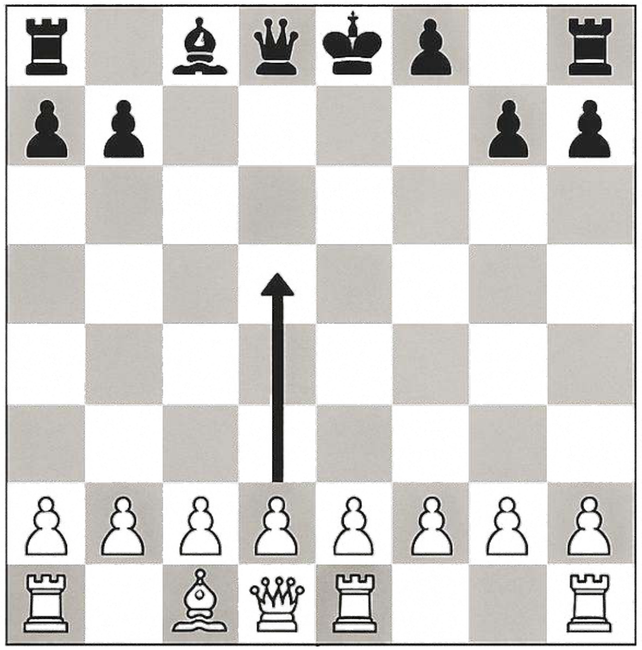
Algorithmic Approaches

Value-Based



Model evaluates each possible position.

Policy Gradient



Model directly outputs next move.

Model-Based Planning

Using a model to compute the optimal next move.

equation 01

Markov Decision Process

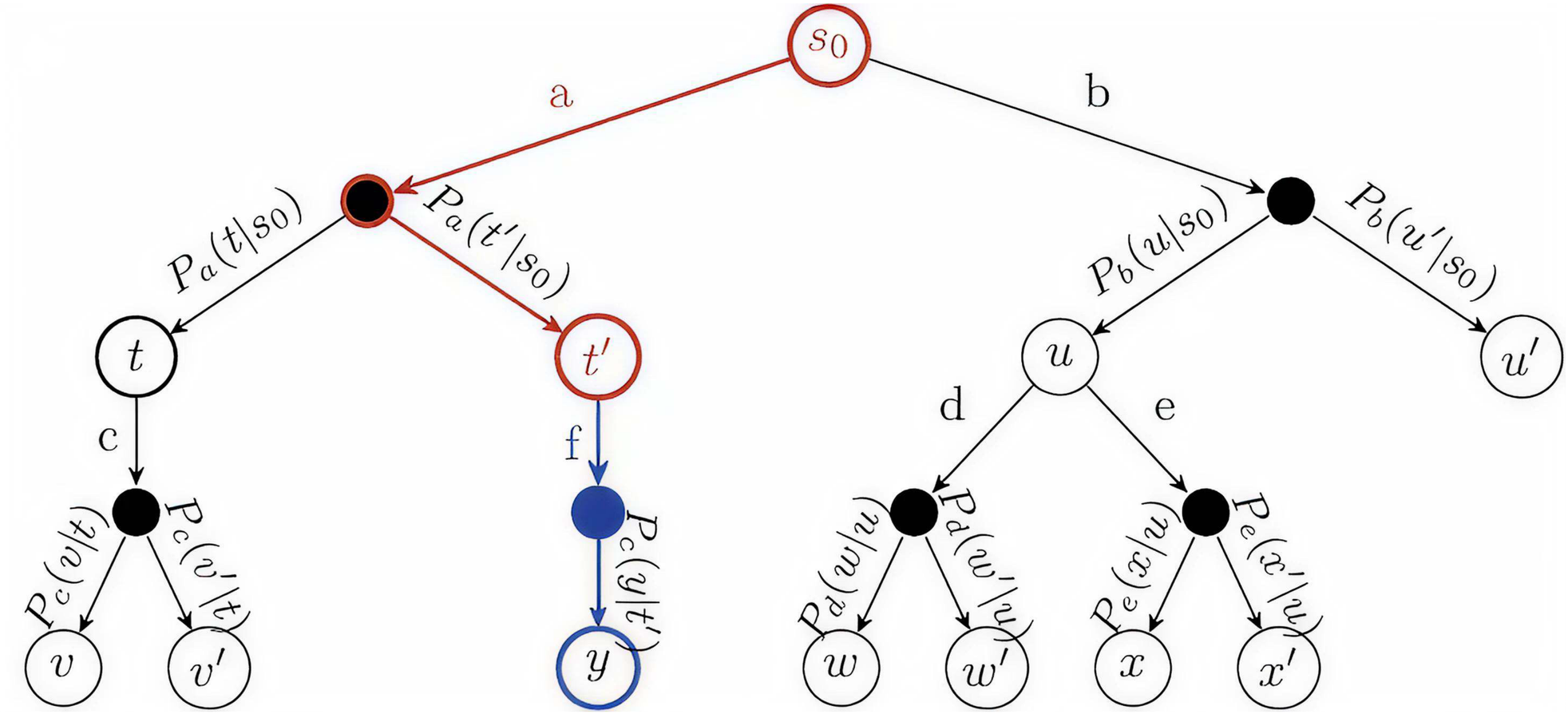
A convention for representing finite environments.

Markov Property	The current state contains all the information needed to predict the future. The future is independent of past states, when given the information contained in the current state.
States - s_t	A list of possible states of the environment.
Actions - a_t	A list of possible actions that can be taken from each state.
Reward Function - r_t	The expected reward received when taking an action in a given state.
Transition Function - P_{ij}	A probability distribution over possible next states, given the current state and an action.
Discount Factor - γ	A discount on the value of future rewards versus present rewards.
Policy - $\pi_\theta(a_t \mid s_t)$	The model that determines what action to take from a given state.

Monte Carlo Tree Search

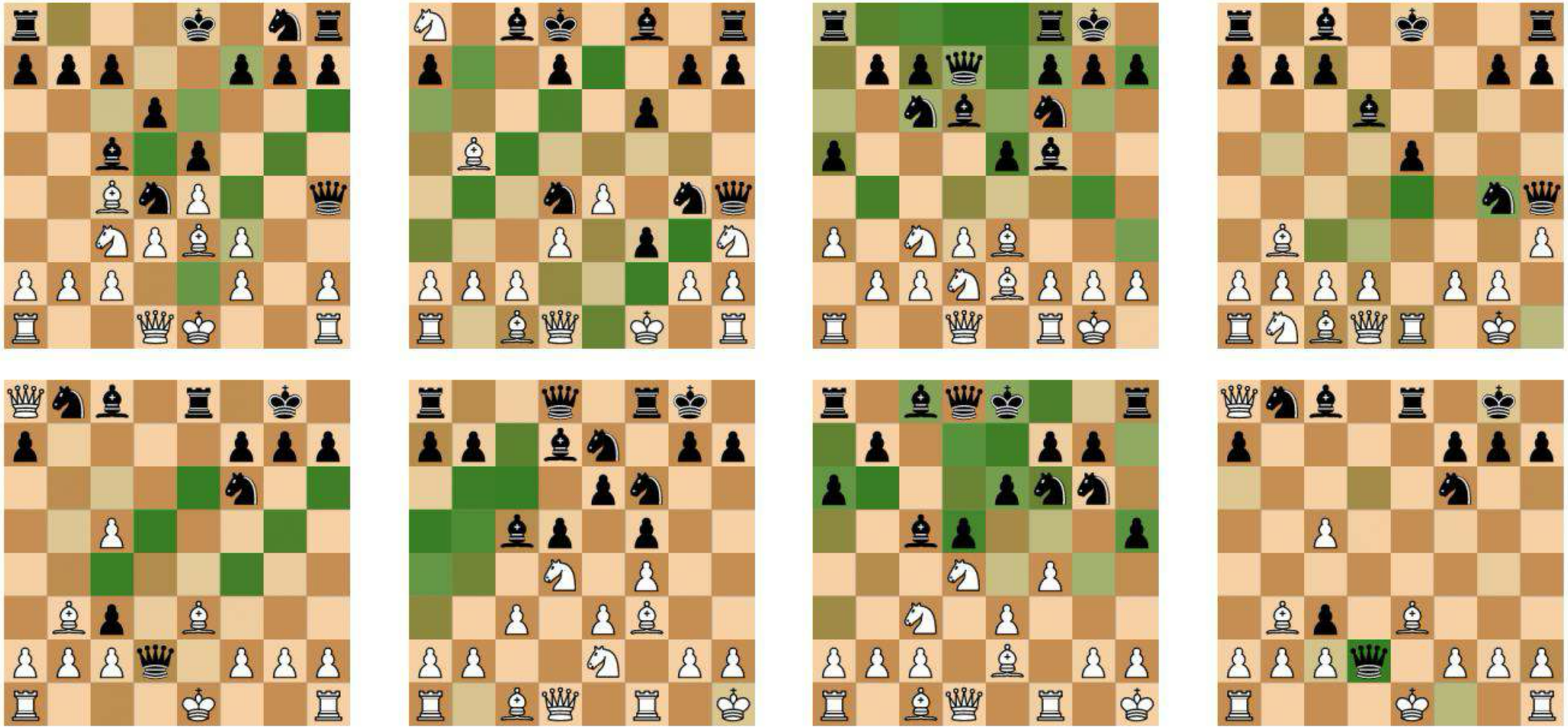
equation 01

Planning with a model of a finite environment



AlphaZero

Using Monte Carlo Tree Search in Intractably Large Environments



AlphaZero

Planning

Selection Choosing the state to explore next in the search tree.

$$a_t = \arg\max_a (Q(s,a) + U(s,a)) \qquad U(s,a) \propto P(s,a)\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}$$

Expansion Adding the next node to the search tree.

Evaluation Running the model to evaluate the new node.

$$P(a|s), v_s = f_\theta(s,a)$$

Propagation Passing the evaluation back up the tree to update node values.

$$N(s,a) \leftarrow N(s,a) + 1 \qquad W(s,a) \leftarrow W(s,a) + v \qquad Q(s,a) = \frac{W(s,a)}{N(s,a)}$$

Inference and Training

For each move in a game:

Loss

$$l = (z-v)^2 - \pi^T log(P) + c||\theta||^2$$

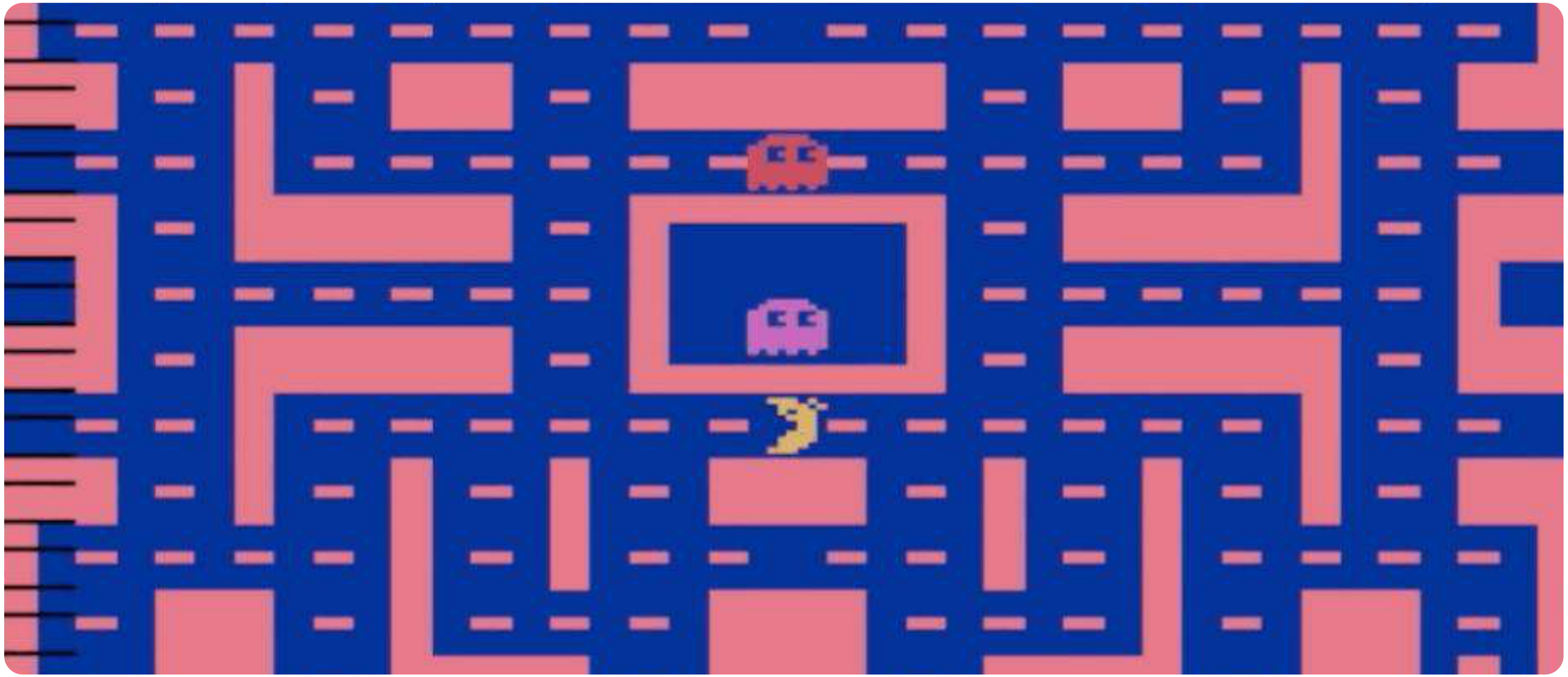
Policy $$\pi(a|s) = \frac{N(s,a)^{1/\tau}}{\sum_b N(s,b)^{1/\tau}}$$

Final Outcome

$$z = \begin{cases} +1, & \text{if the player wins,} \\ 0, & \text{if the game is a draw,} \\ -1, & \text{if the player loses.} \end{cases}$$

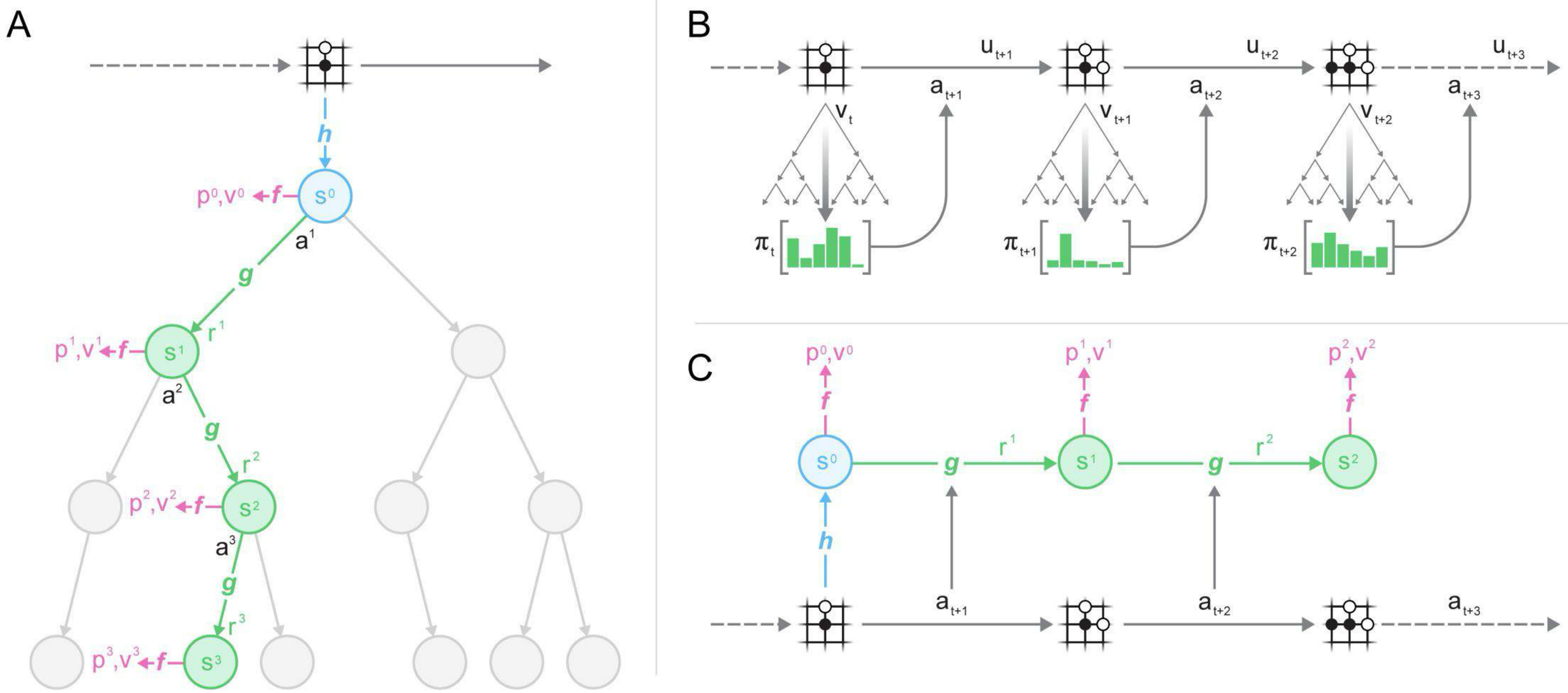
MuZero

Scaling Up Monte Carlo Tree Search to Larger Spaces



MuZero

Learning a latent space model of the environment



MuZero

Learning a latent space model of the environment

Dynamics Function

$$r_k, s_k = g_\theta\left(s_{k-1}, a_k\right)$$

Designed to be “Value Equivalent”. Predicts the next latent state and is trained to match the reward from the real environment.

Prediction Function

$$p_k, v_k = f_\theta\left(s_k\right)$$

Predicts the next policy and value for a given state.

Representation Function

$$s^0 = h_\theta\left(o_1, \ldots, o_t\right)$$

Converts raw observations into an initial hidden state. Obeys the Markov property by embedding history into that state.

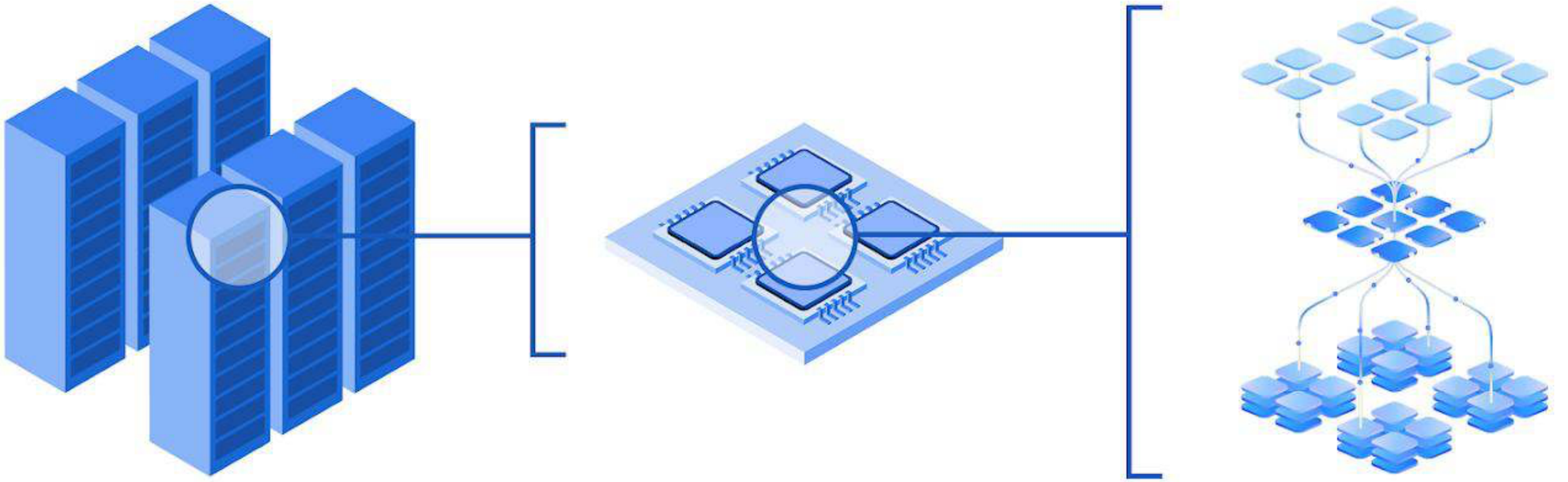
Loss

$$\ell_t(\theta) = \sum_{k=0}^{K}\left[l_r(u_{t+k}, r_t^k) + l_v(z_{t+k}, v_t^k) + l_p(\pi_{t+k}, p_t^k)\right] + c\|\theta\|^2$$

Converts raw observations into an initial hidden state. Obeys the Markov property by embedding history into that state.

AlphaEvolve

Searching in the space of engineering problems



Data Center Optimization
Borg Scheduling

Hardware Optimization
TPU Circuit Design

Software Optimization
Gemini Training