

Reinforcement Learning

LMARL Session 2, February 18th 2025, Polina Tsvilodub

In part based on slides of Michael Franke (SS 2018)

Recap: Cognitive architecture

“[A] particular methodology for building [agents]. It specifies how . . . the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions . . . and future internal state of the agent. An architecture encompasses techniques and algorithms that support this methodology” (Maes, 1991, p.115)

Kaelbling considers an agent architecture to be:

“[A] specific collection of software (or hardware) modules, typically designated by boxes with arrows indicating the data and control flow among the modules. A more abstract view of an architecture is as a general methodology for designing particular modular decompositions for particular tasks.” (Kaelbling, 1991, p.86)

cognitive architectures are both a functional description and a theory of humanlike minds, designed to model such minds and their functionality leading to intelligence

Cognitive architecture

Overview of general components:

- ▶ **Perception:** transform raw sensory input into representations
- ▶ **Attention Mechanisms:** allocate cognitive resources to certain information
- ▶ **Action Selection:** decision-making processes as to which actions to undertake
- ▶ **Memory Systems:**
 - Short-term memory
 - Working Memory: "temporary storage" for active tasks
 - Long-Term Memory: "permanent" storage for knowledge, rules, and experiences
 - episodic memory
 - procedural memory
 - semantic memory
- ▶ **Learning:**
 - adaptation, generalization based on experiences
- ▶ **Reasoning & Metacognition:**
 - Logical inference, probabilistic models, DPT, self-reflective thinking

General Problem Solver

Means-ends analysis

- ▶ define problem (current and goal state), **specification of pre- and post-conditions of actions**
- ▶ compare current and goal states
- ▶ identify differences
- ▶ select an operator to reduce the difference
- ▶ apply the operator and repeat until goal is reached

What are the assumptions?

- ▶ structured state and action representations
 - ▶ respecified set of goal types and operators
-
- ▶ **Problem solving as a search over structured problem space**

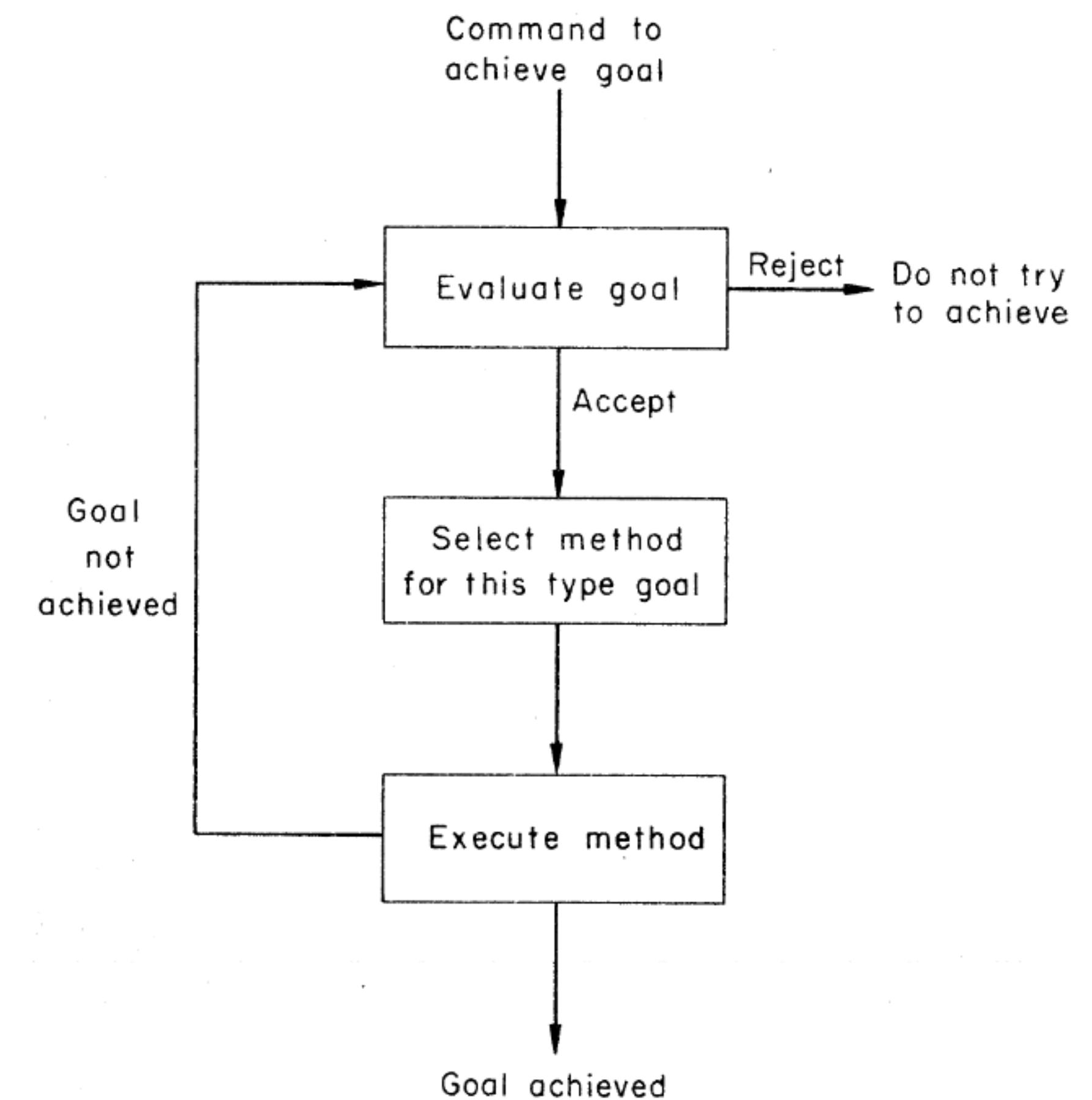
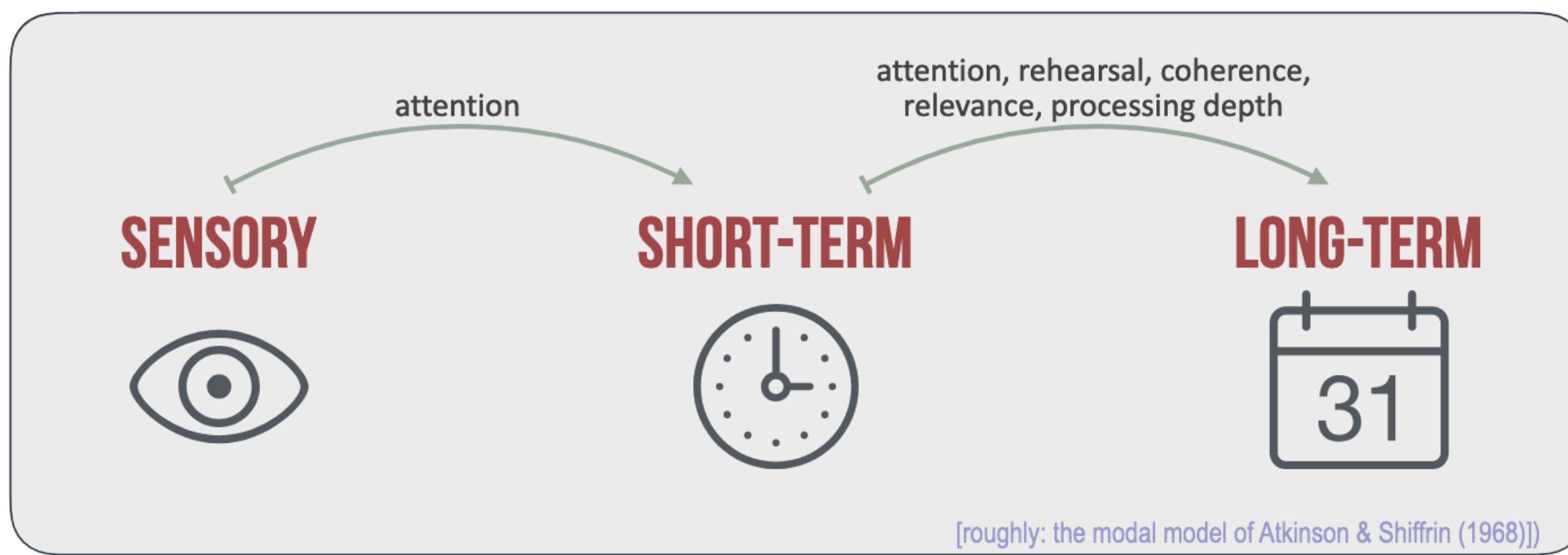


Fig. 1—Executive organization of GPS

Learning

Based on memory

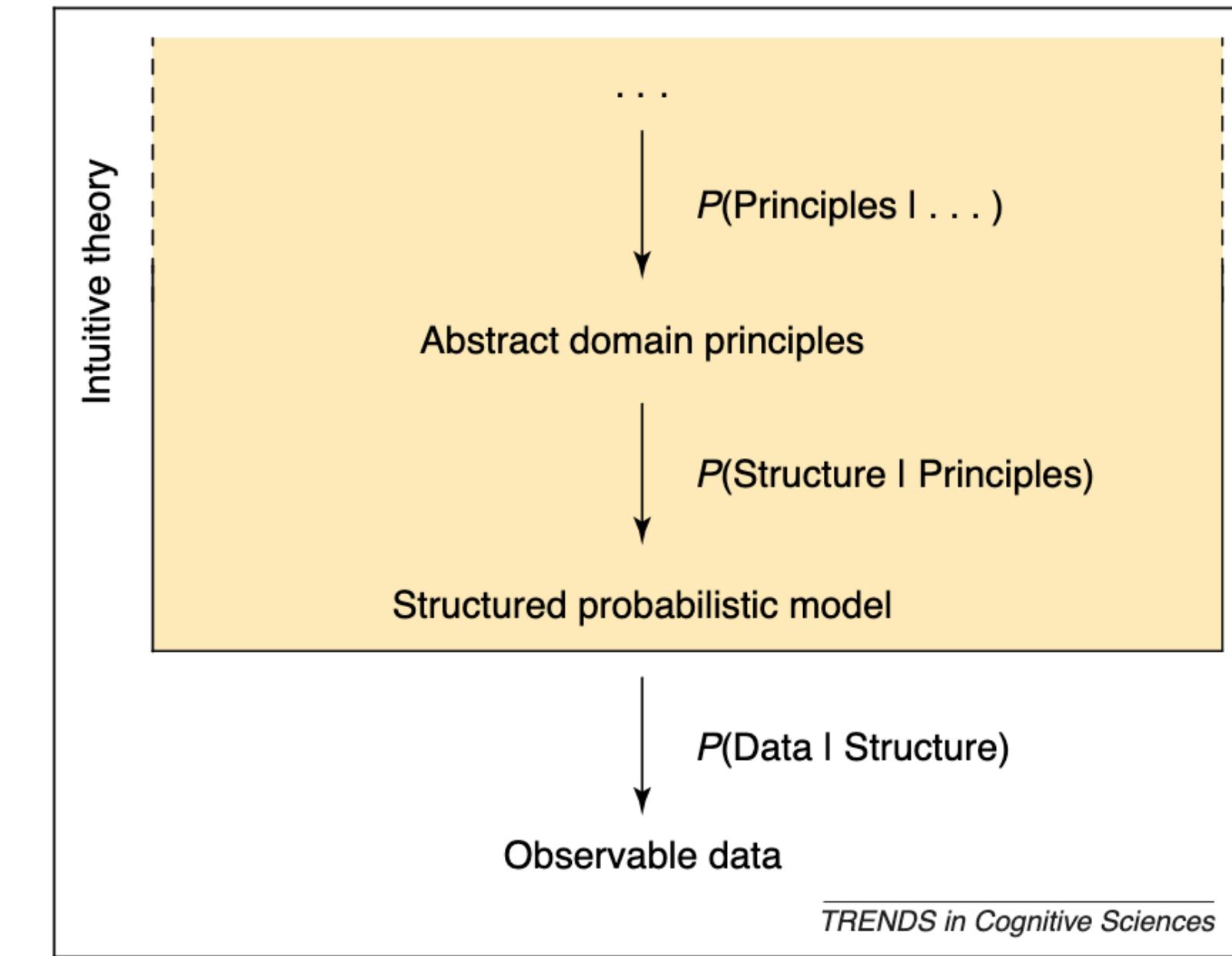
- ▶ associative vs. non-associative
- ▶ explicit and implicit
- ▶ transfer learning
- ▶ social learning



Bayesian models of inductive learning



$$P(h|x, T) = \frac{P(x|h, T)P(h|T)}{\sum_{h' \in H_T} P(x|h', T)P(h'|T)}$$



Flavors of machine learning

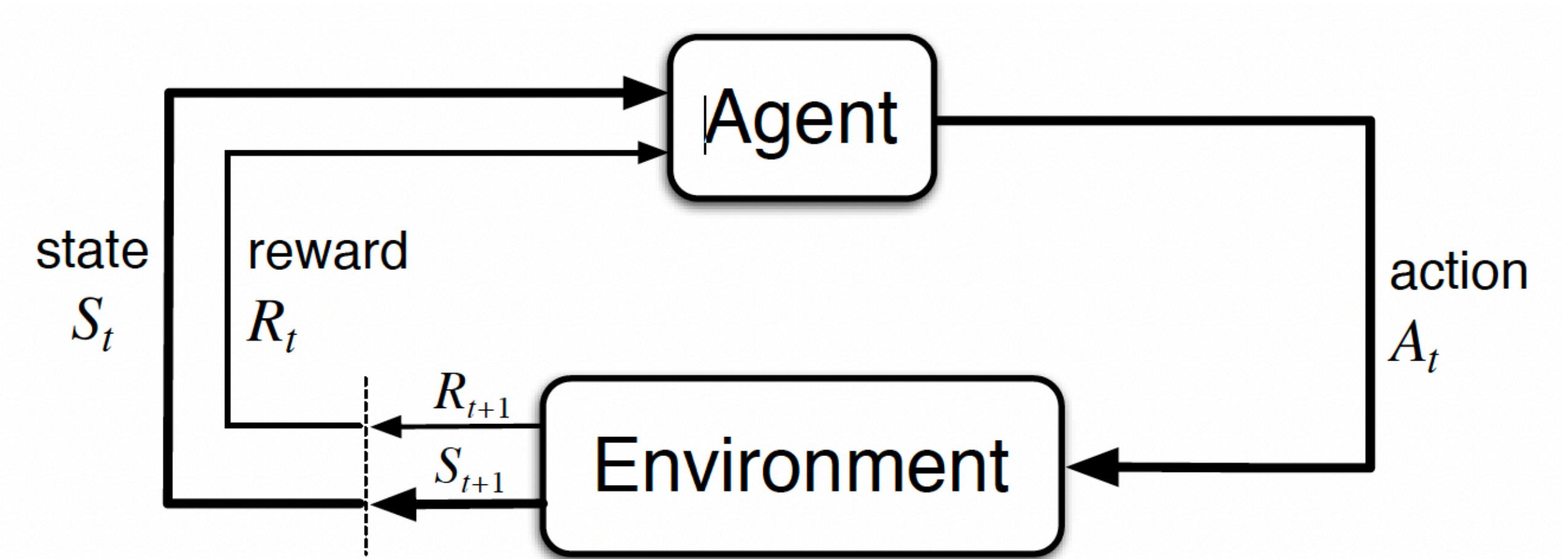
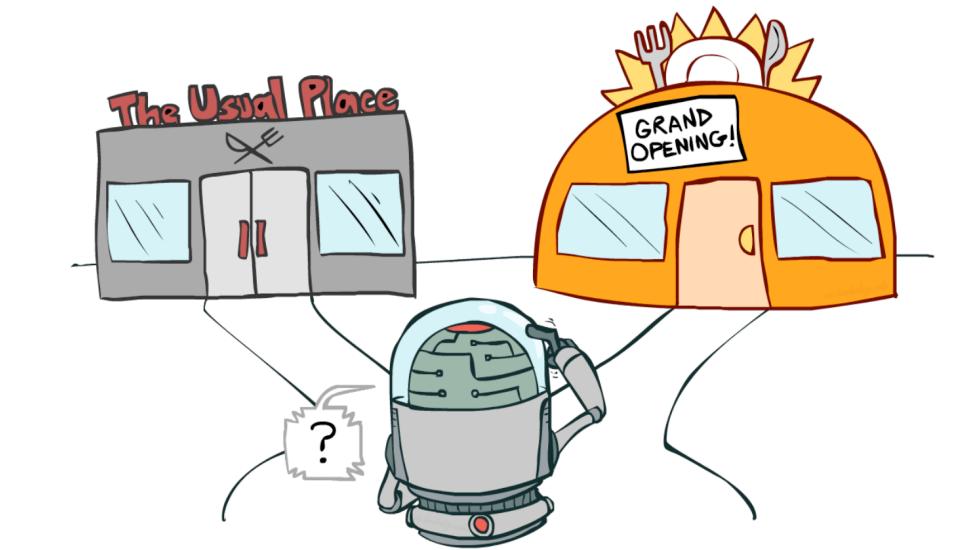
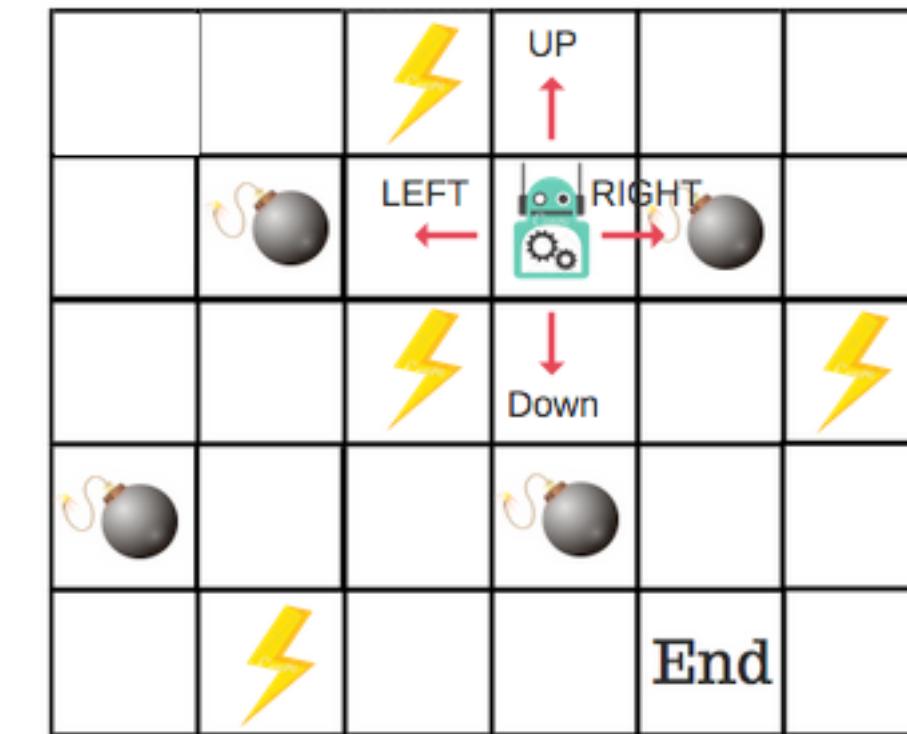
Roughly: Learn ‘intelligent’ behavior from data so as to be able to generalise to new situations

- ▶ Unsupervised learning
 - e.g., clustering
 - ▶ discover patterns in unlabeled data
 - ‘given my inductive bias, what is the likely structure of the data?’
 - ▶ Supervised learning
 - also self-supervised learning
 - aka behavioural cloning
 - ▶ learn to output Y, given X, from labeled data
 - ‘do as I show you’
 - ▶ learning from demonstration
- ▶ Reinforcement learning
 - trial-and-error learning
 - ▶ learning from interaction / experience
 - ‘how do I optimally behave in order to maximize reward?’
 - **reward hypothesis**
 - or, ‘how do i optimally achieve my goal?’
 - most natural way of learning?
 - tightly connected to the way organisms behave (“pleasure maximizers”)

Reinforcement Learning: Overview

Introduction

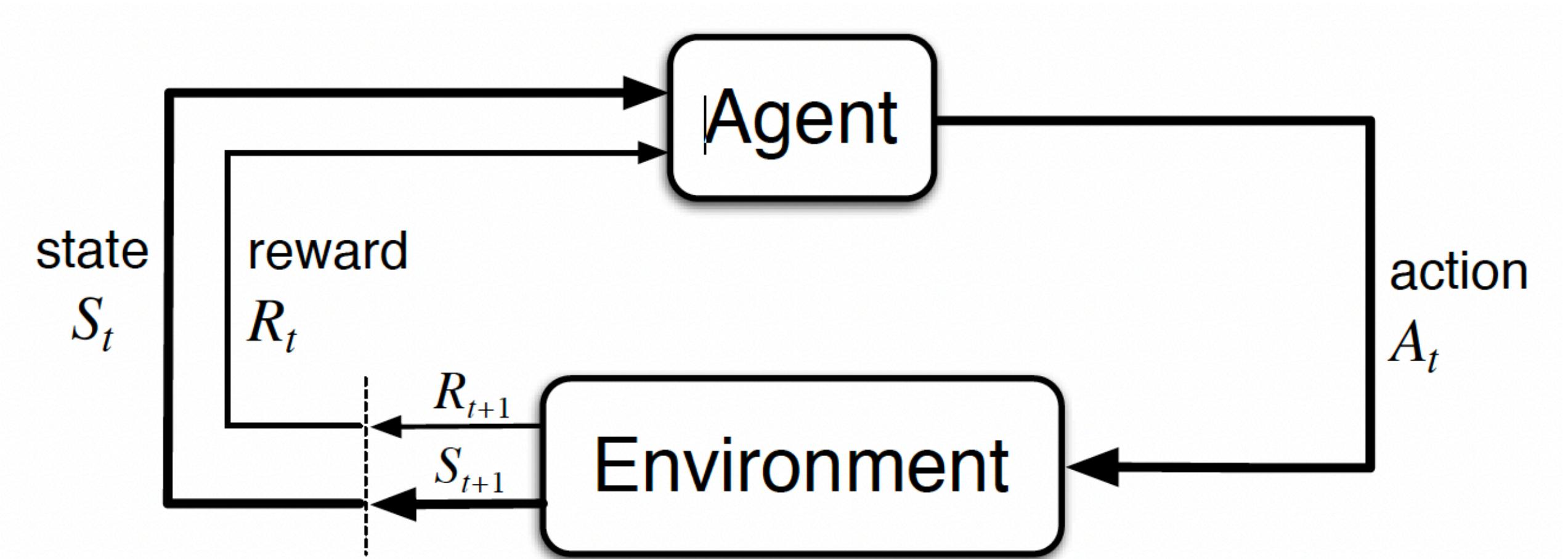
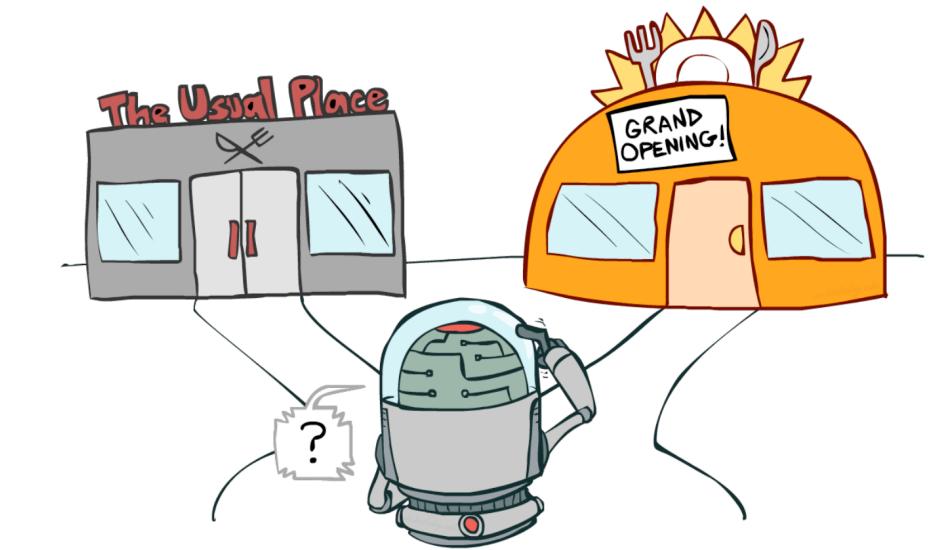
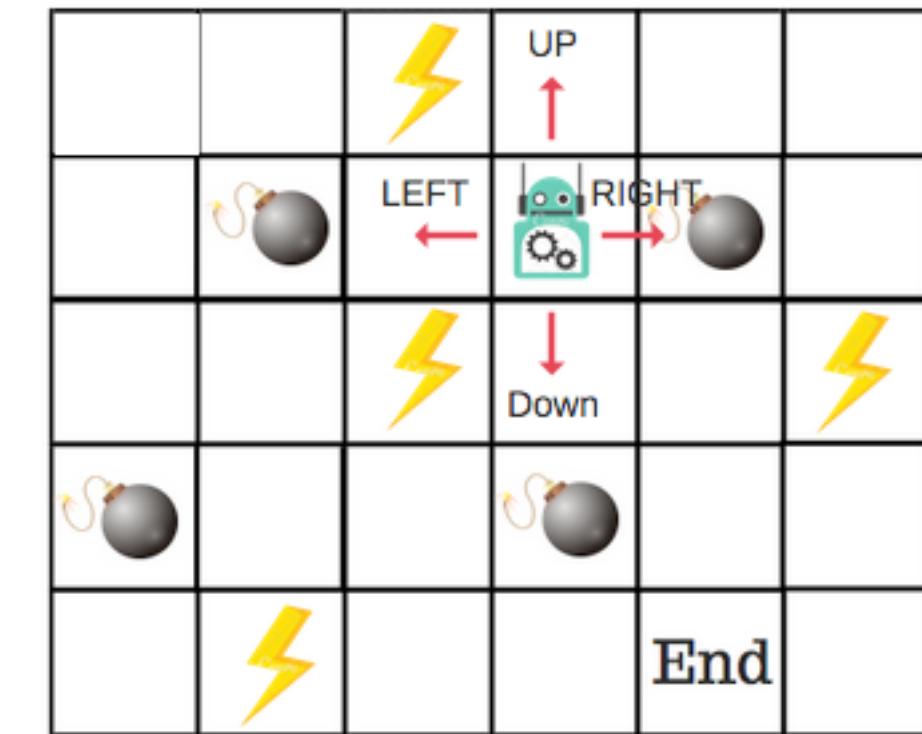
- ▶ **Reinforcement Learning:** Computational formalisation of goal-directed learning and decision making
- ▶ **Goal:** Maximize rewards
(by learning optimal behavior)
- ▶ **Basic building blocks:**
 - Agent
 - States
 - Actions
 - Transition function P
 - Reward
 - Policy



Reinforcement Learning: Overview

Introduction

- ▶ **Reinforcement Learning:** Computational formalisation of goal-directed learning and decision making
- ▶ **Goal: Maximize rewards**
(by learning optimal behavior)
- ▶ **Basic building blocks:**
 - Agent
 - States: $S_t \in S$ for $t = 0, 1, 2, 3, \dots$
 - Actions: $A_t \in A(s)$
 - Transition function $P(s' | s, a)$
 - Reward: $R_{t+1} \in R$
 - Policy: $\pi(S_t) = P(A_t | S_t)$



Markov Decision Processes

Formal definition

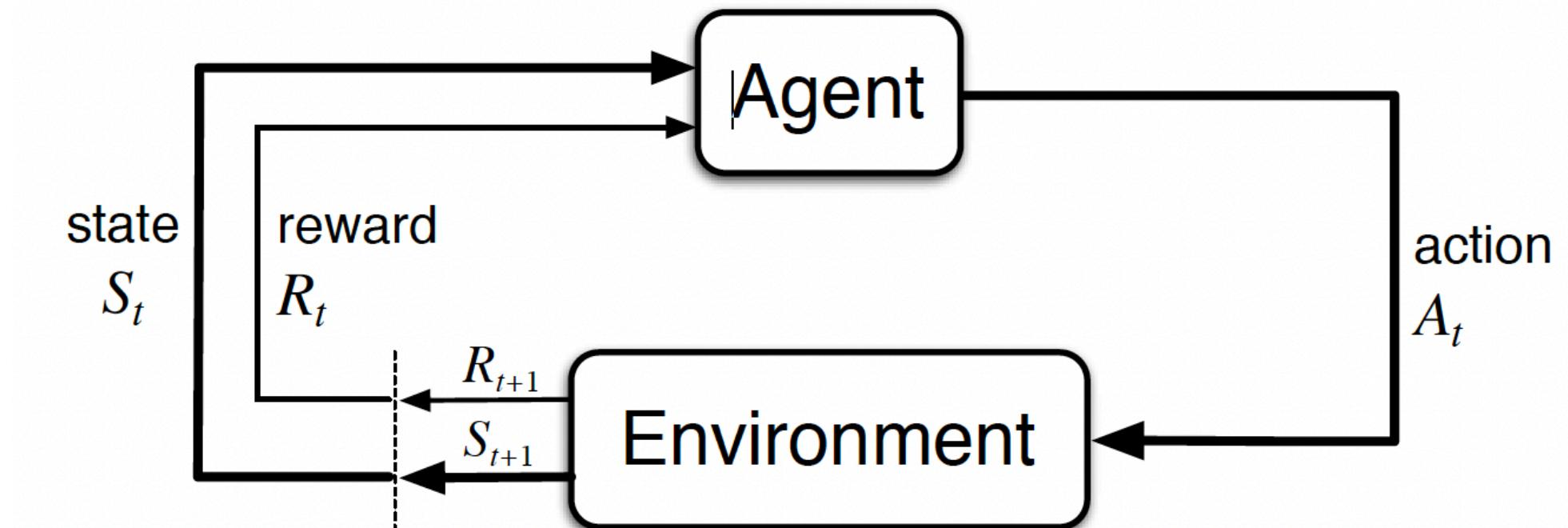
Finite MDPs: (S, A, T, R)

describes sequential decision making $(s_0, a_0, R_1, s_1, a_1, R_1, \dots)$

1. $S_t \in S$ for $t = 0, 1, 2, 3, \dots$
2. $A_t \in A(s)$
3. $R_{t+1} \in R$
4. $T(s' | s, a) = \sum_{r' \in R} P(s', r' | s, a)$
5. Horizon H , discount factor $0 \leq \gamma \leq 1$

Expected rewards for state s and action a : $r(s, a) = \mathbb{E}(R_t | S_{t-1} = s, A_{t-1} = a) = \sum_{r \in R} \sum_{s' \in S} r P(s', r | s, a)$

Markov property: S_{t-1}, A_{t-1} include all information about past agent-environment interactions that are relevant for S_t, R_t

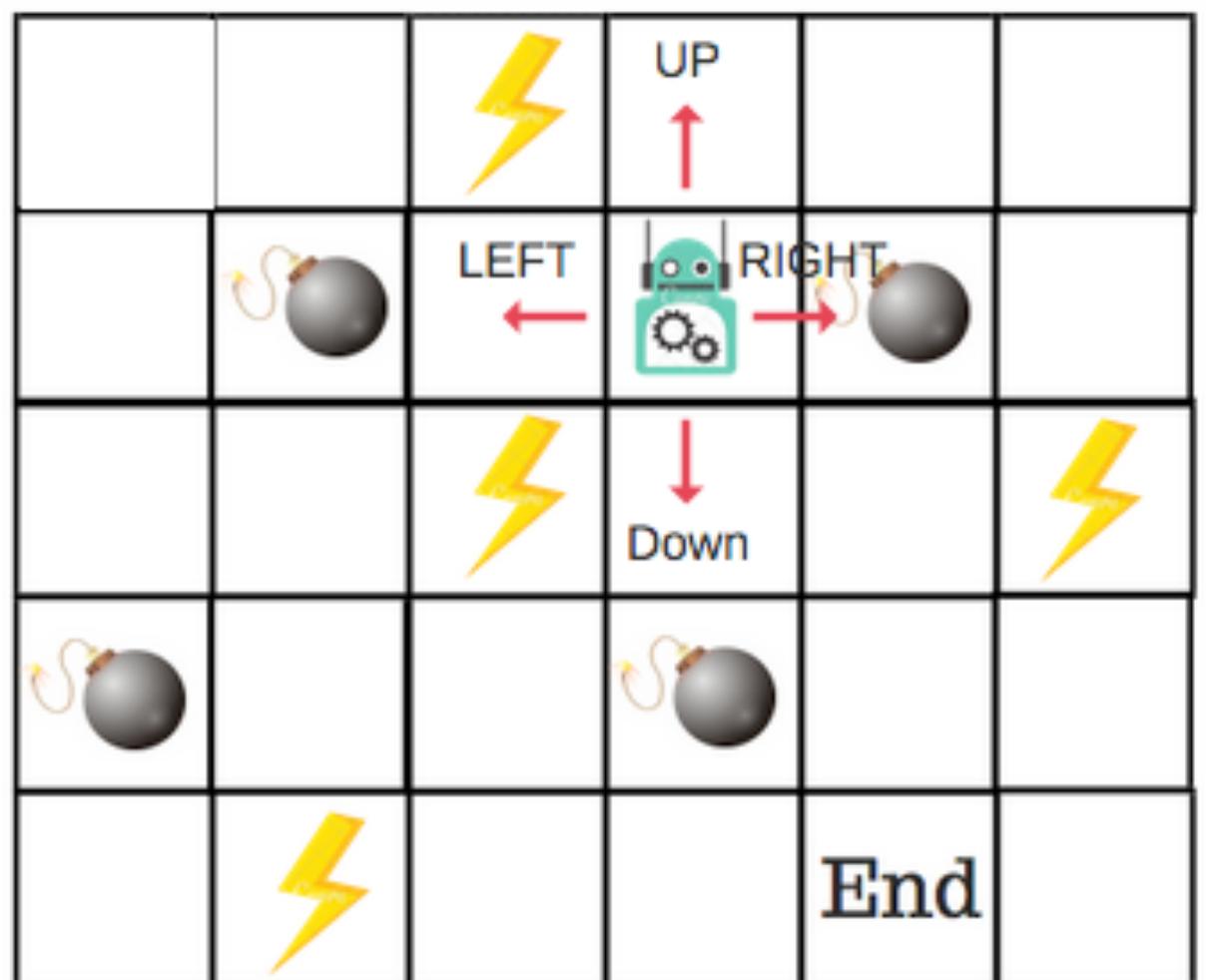


Markov Decision Processes

Formal definition

Finite MDPs: (S, A, T, R)

1. $S_t \in S$ for $t = 0, 1, 2, 3, \dots$
2. $A_t \in A(s)$
3. $R_{t+1} \in R$
4. $T(s'|s, a) = \sum_{r' \in R} P(s', r'|s, a)$



Goal: maximize **returns** until goal achieved

$$G_t = R_{t+1} + R_{t_2} + \dots + R_T$$

Formally: maximize expected **discounted** rewards over **episode**

$$G_t = R_{t+1} + \gamma R_{t_2} + \gamma^2 R_{t+3} + \dots + \gamma^{T-t-1} R_T = \sum_{k=t+1}^T \gamma^{k-t-1} R_k$$

Markov Decision Processes

Optimization Problem

- ▶ **Goal:** Maximize accumulated rewards (=returns): $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

- ▶ **Basic building blocks:**

- Agent
- States: $S_t \in S$ for $t = 0, 1, 2, 3, \dots$
- Actions: $A_t \in A(s)$
- Reward: $R_{t+1} \in R$
- Policy: $\pi(S_t) = P(A_t | S_t)$

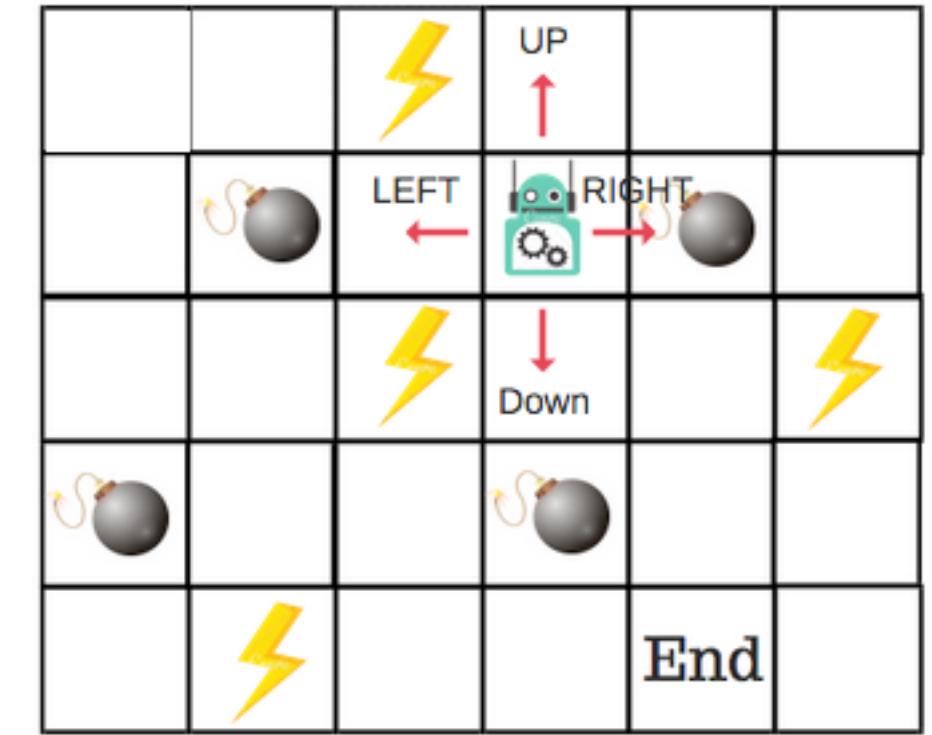
- ▶ We can identify optimal way to behave if we know what good particular states and/or actions are:

State-value function: $v_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right]$ for all s
think: "How good is it to be in state s ?"

Action-value function: $q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$ for all s, a
think: "How good is it to take action a in state s ?"

- ▶ **Can be estimated from experience!**

- ▶ Optimal policy π^* : $\pi \geq \pi' \Leftrightarrow v^*_{\pi^*}(s) \geq v_{\pi}(s)$ for all s and $q^*_{\pi^*}(s, a) = \max_{\pi'} q_{\pi'}(s, a)$



Markov Decision Processes

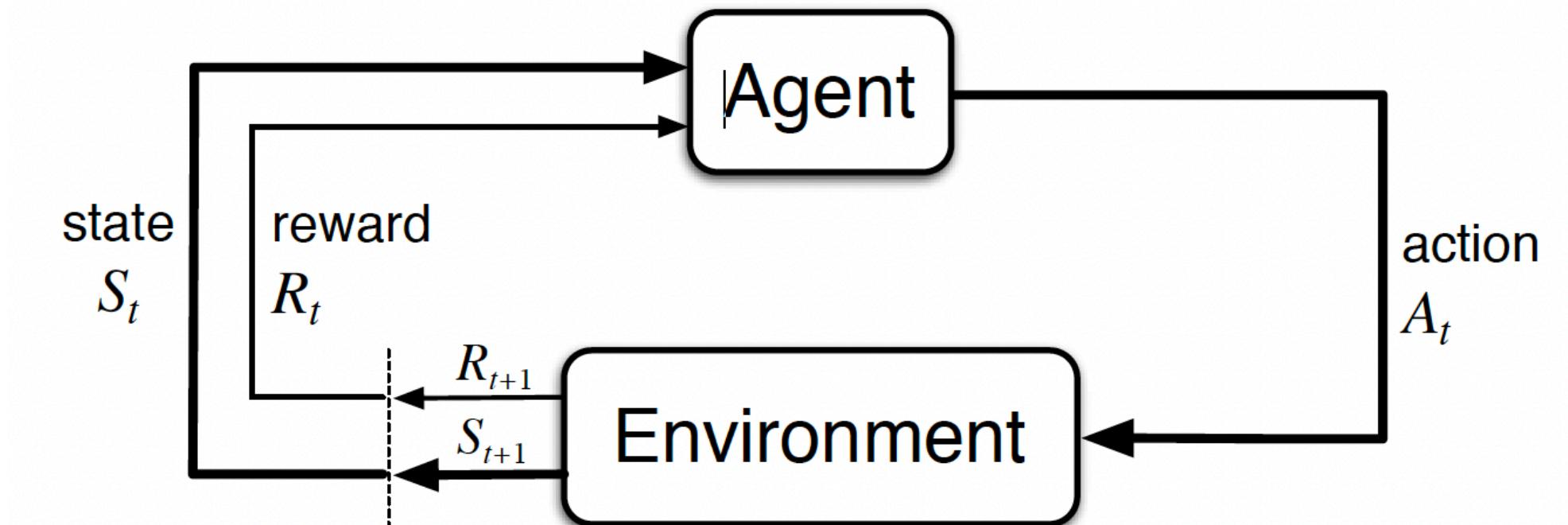
Formal definition

Finite MDPs: (S, A, T, R)

1. $S_t \in S$ for $t = 0, 1, 2, 3, \dots$
2. $A_t \in A(s)$
3. $R_{t+1} \in R$
4. $T(s'|s, a) = \sum_{r' \in R} P(s', r'|s, a)$
5. Horizon H , discount factor $0 \leq \gamma \leq 1$

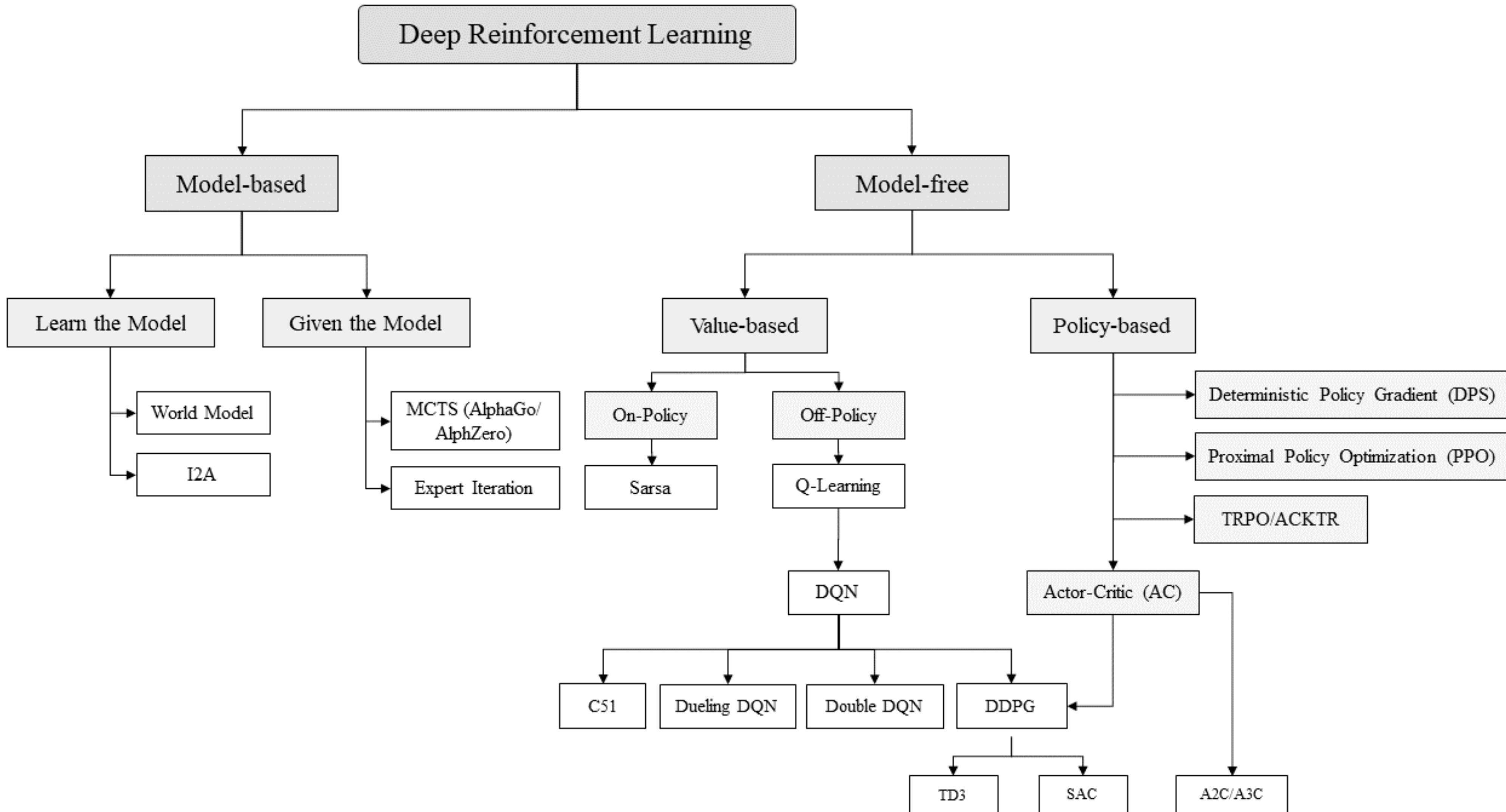
Expected rewards for state s and action a : $r(s, a) = \mathbb{E}(R_t | S_{t-1} = s, A_{t-1} = a) = \sum_{r \in R} \sum_{s' \in S} r P(s', r | s, a)$

Markov property: S_{t-1}, A_{t-1} include all information about past agent-environment interactions that are relevant for S_t, R_t



RL Algorithms

Approximating Optimal Policy



RL Agents and Intelligent Agents

How do CAs, notions of agency, RL relate?

- ▶ RL agents focus on modeling learning through trial-and-error, while agents more generally have many more components
- ▶ model-based RL agents might be seen as closer to representing the world & planning
 - Q: what kind of memory is an RL policy similar to?
- ▶ exploration-exploitation problem in RL
- ▶ are RL systems agents in intentional sense / sense of Wooldridge & Jennings?
 - Maybe: yes in the narrow sense of learning goal-directed behavior
 - No in the broader sense of having updatable beliefs, dynamic intentions, reasoning, being autotelic
- ▶ there is work on building intrinsically motivated RL agents



LLMs as RL agents

Language Modeling ≠ Assisting Users

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

Language models are not *aligned* with user intent [[Ouyang et al., 2022](#)].

Making LLMs useful

Adaptation

- ▶ training a task-specific head on top of a model
 - e.g., span prediction layer on top of BERT with frozen BERT
 - on a dataset of ground truth input-output pairs for a particular task
- ▶ fine-tuning the model
 - further training part or entire model for a shorter time
 - on a dataset of ground truth input-output pairs for a particular task
- ▶ practical problem
 - training with standard supervision is impractical (data collection)
 - and inefficient (restricting “ground truth” to finite set of answers for open-ended tasks)
- ▶ **RL is the solution:** learn to achieve goal based on feedback from environment rather than direct demonstration of correct behaviour

Language model

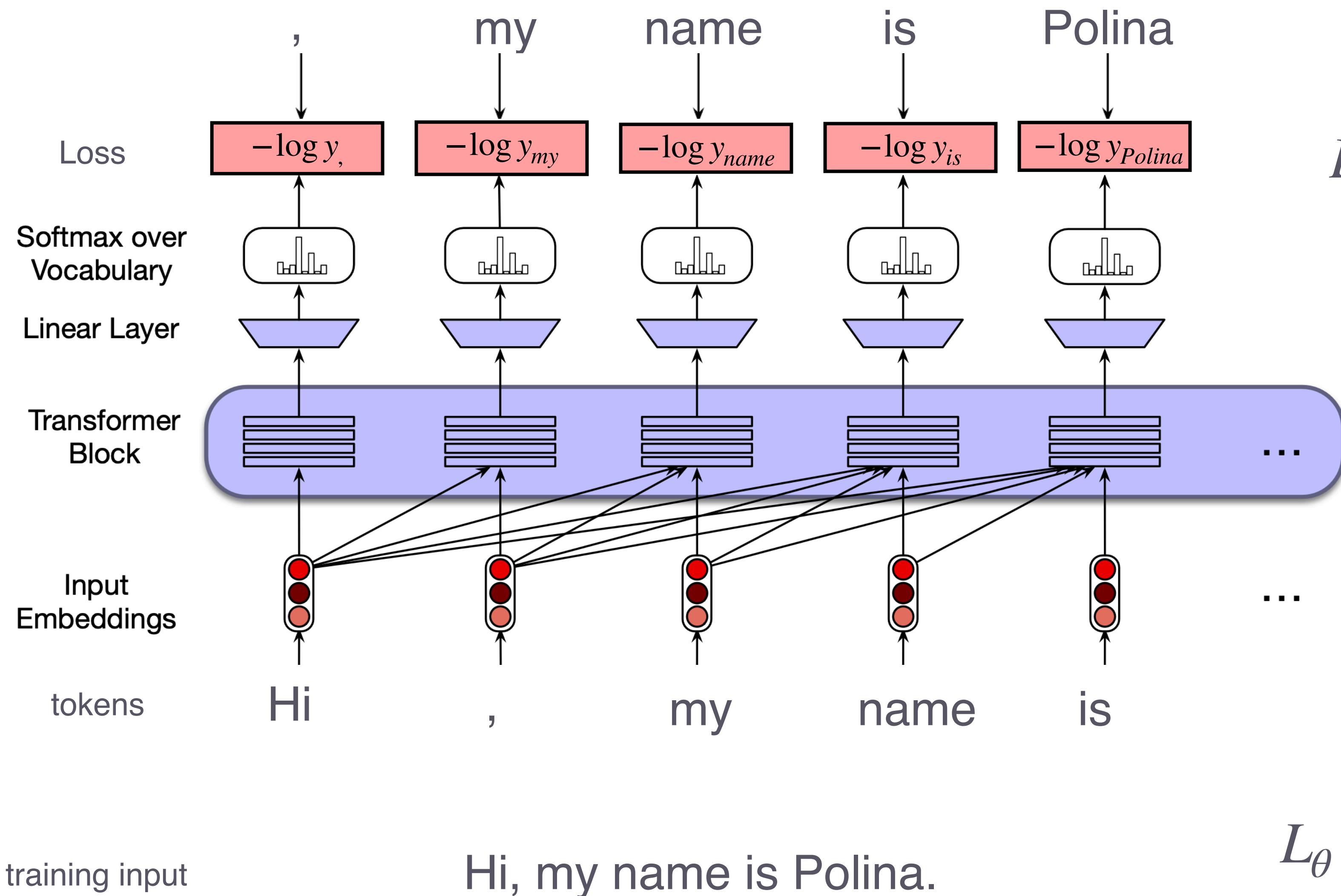
left-to-right / causal model

- let \mathcal{V} be a (finite) **vocabulary**, a set of words
 - we say “words” but these can be characters, sub-words, units ...
- let $w_{1:n} = \langle w_1, \dots, w_n \rangle$ be a finite sequence of words
- a **causal language model** is defined as a function that maps an initial sequence of words to a probability distribution over words:
 $LM : w_{1:n} \mapsto \Delta(\mathcal{V})$
 - we write $P_{LM}(w_{n+1} \mid w_{1:n})$ for the **next-word probability**
 - the **surprisal** of w_{n+1} after sequence $w_{1:n}$ is $-\log(P_{LM}(w_{n+1} \mid w_{1:n}))$

Maximizing next-token probability

RECAP

Pretraining



Cross-entropy loss:

$$L_\theta = - \sum_i^{|V|} Q(y_i) \log P(y_i)$$

$$L_\theta = - \log P(y_i)$$

$$L_\theta = \frac{1}{n} \sum_i^n - \log P(y_i)$$

$$L_\theta = \frac{1}{b} \sum_j^b \frac{1}{n_j} \sum_i^{n_j} - \log P(y_{ji})$$

Language Modeling ≠ Assisting Users

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION **Human**

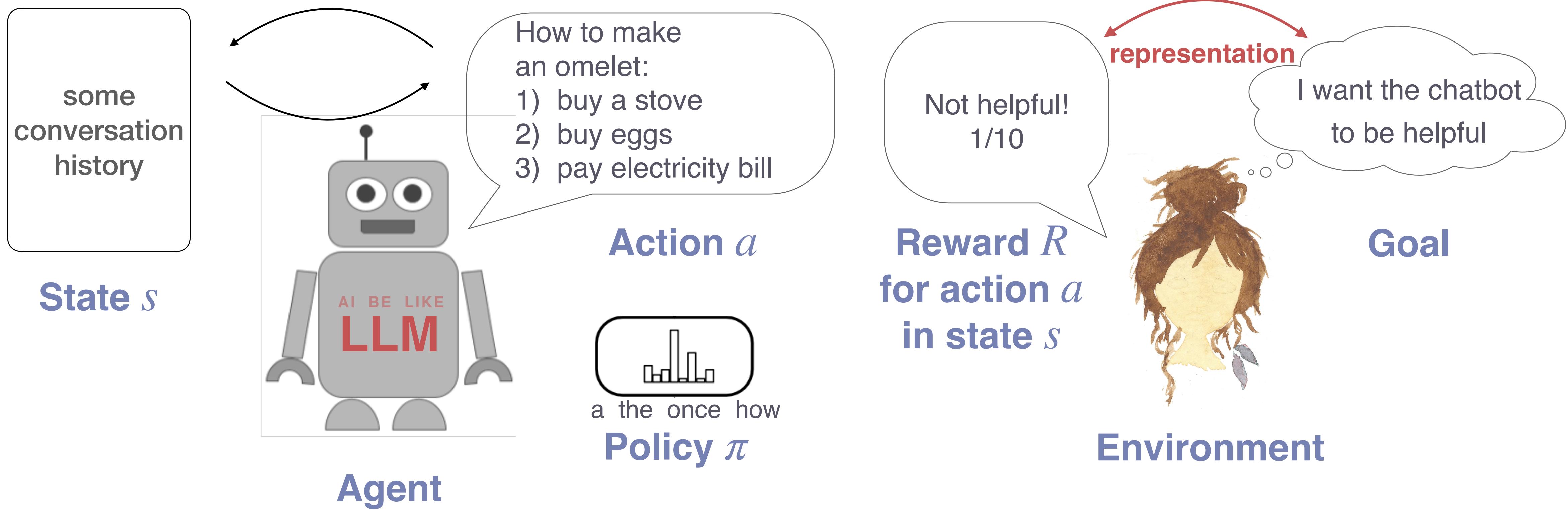
A giant rocket ship blasted off from Earth carrying astronauts to the moon. The astronauts landed their spaceship on the moon and walked around exploring the lunar surface. Then they returned safely back to Earth, bringing home moon rocks to show everyone.

Language models are not *aligned* with user intent [[Ouyang et al., 2022](#)].
Finetuning to the rescue!

Reinforcement Learning from Human Feedback

Overview

- ▶ use human judgments as a signal on what model prediction counts as a good output
- ▶ learn a **reward model representing human feedback**



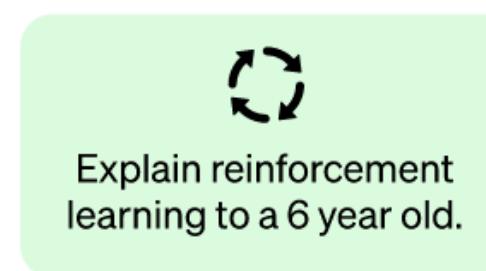
Human feedback in RL

RLHF

Step 1

Collect demonstration data and train a supervised policy.

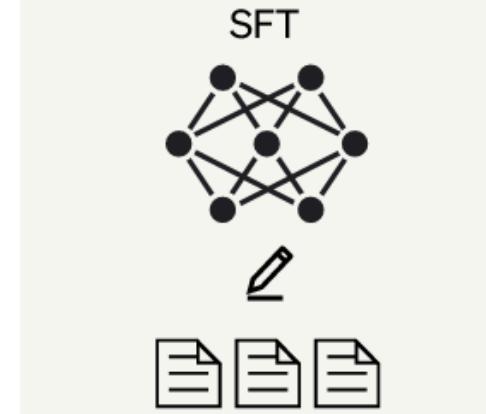
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



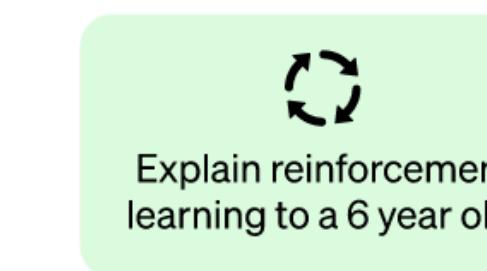
This data is used to fine-tune GPT-3.5 with supervised learning.



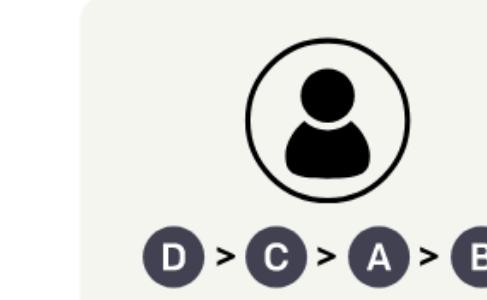
Step 2

Collect comparison data and train a reward model.

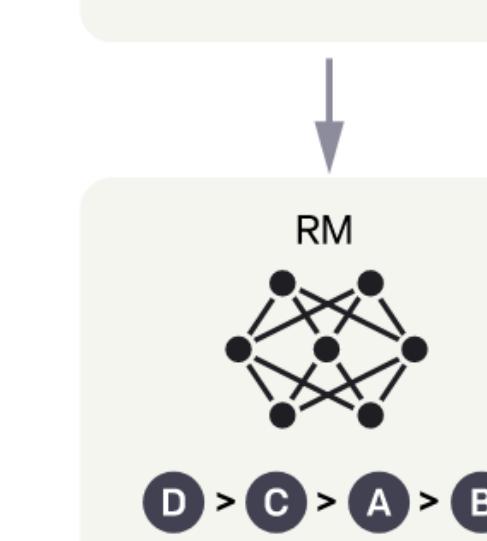
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



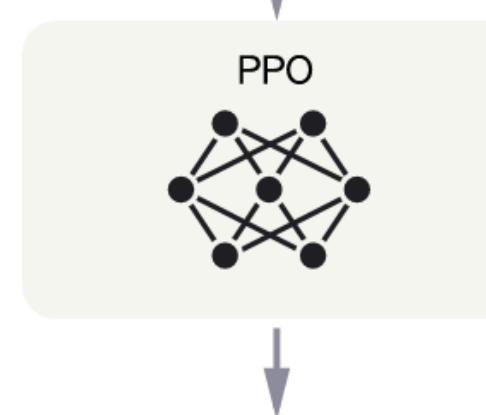
Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

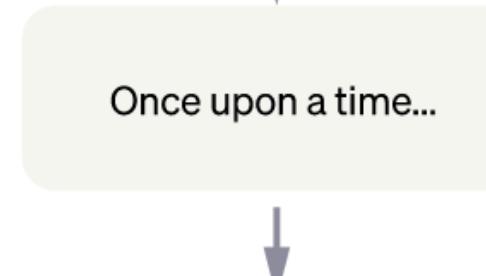
A new prompt is sampled from the dataset.



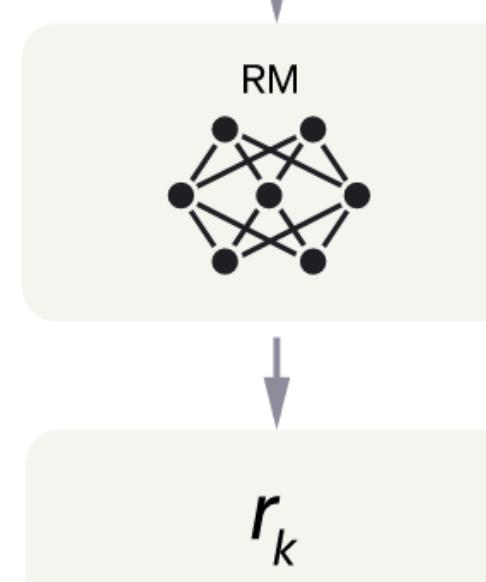
The PPO model is initialized from the supervised policy.



The policy generates an output.



The reward model calculates a reward for the output.



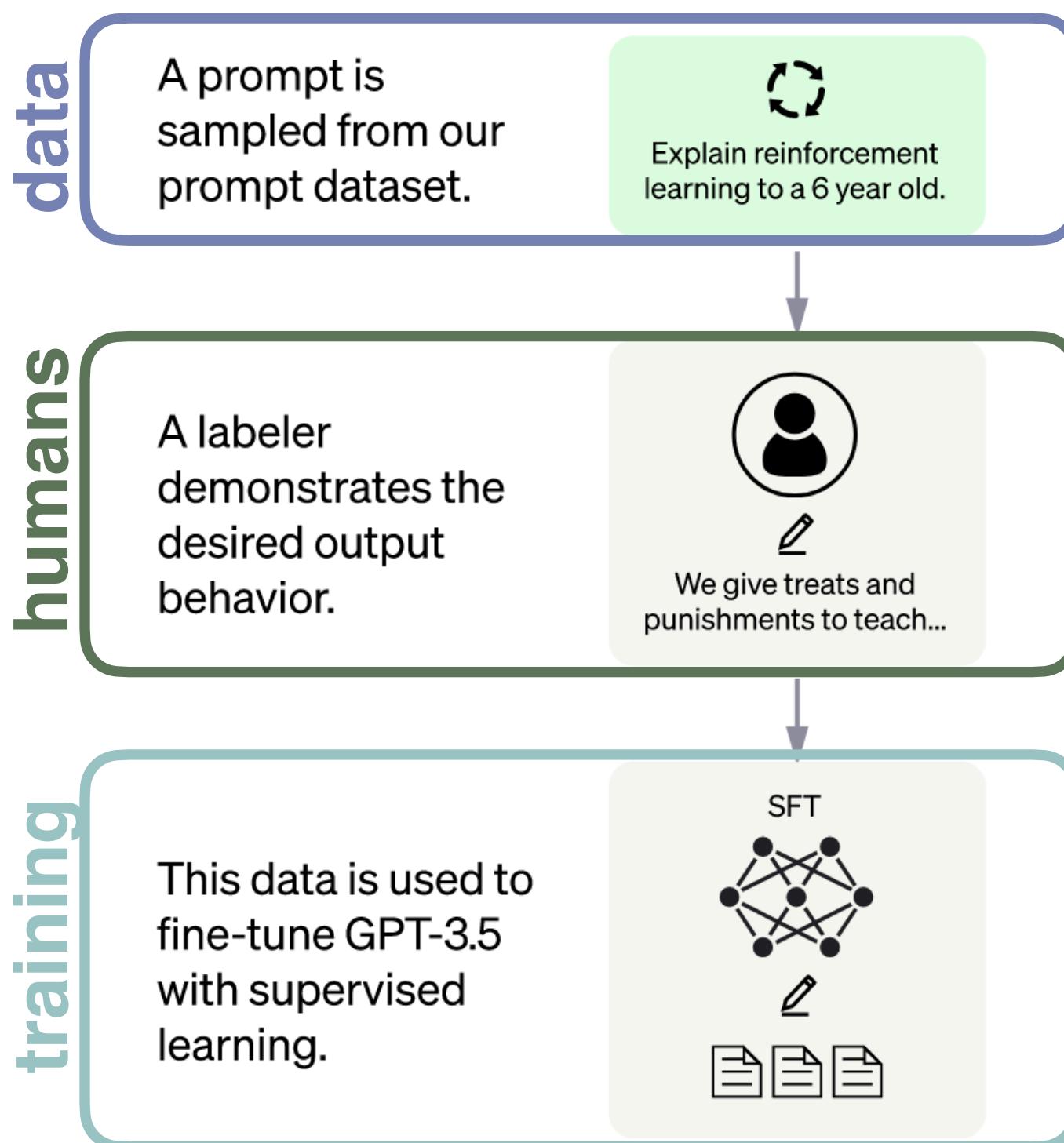
The reward is used to update the policy using PPO.

RLHF in practice

Step 1

Step 1

Collect demonstration data
and train a supervised policy.

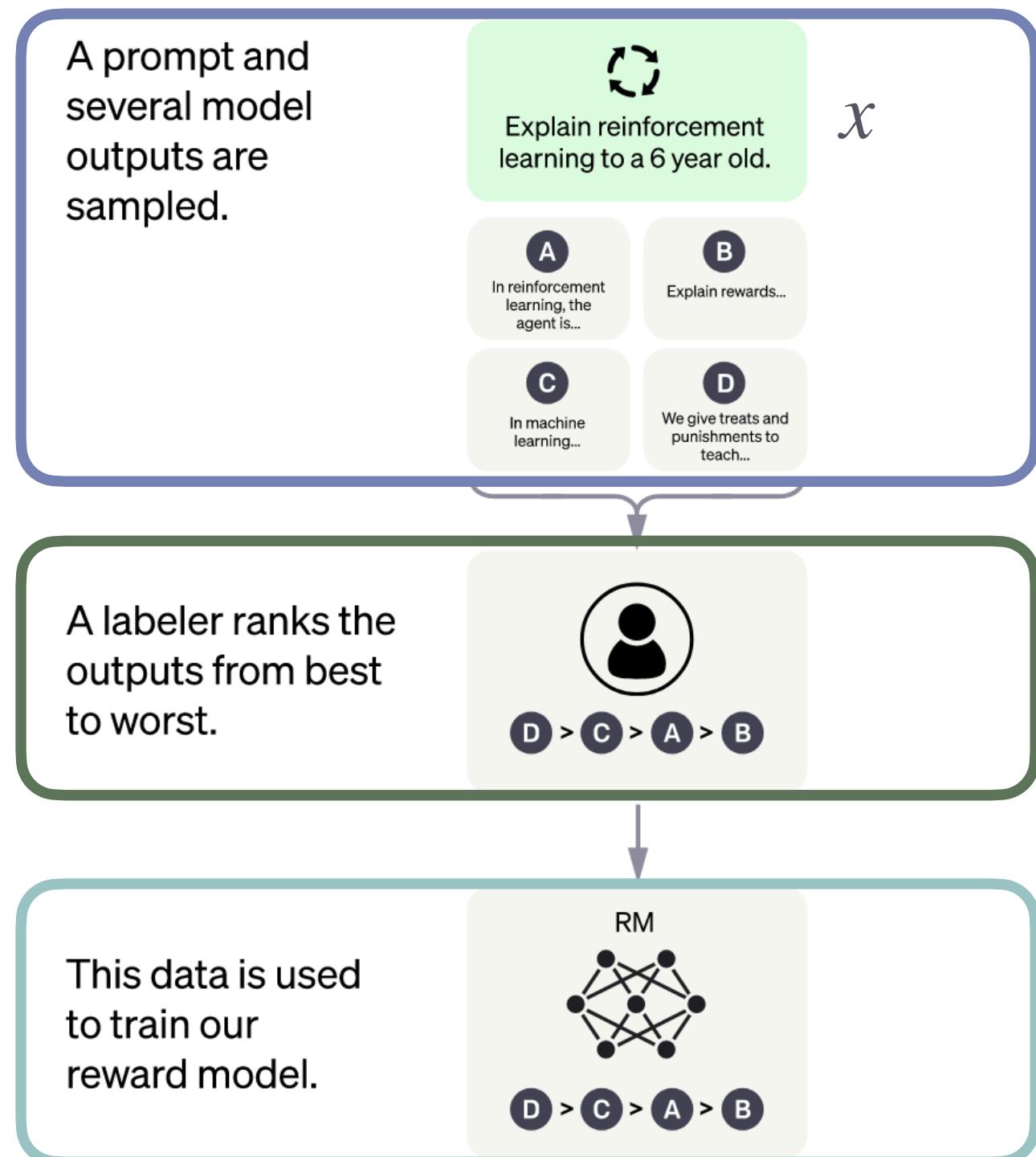


- ▶ supervised fine-tuning on a dataset of input-output demonstrations of the target task
 - pretrained model trained for a shorter time
- ▶ shifts the initial pretraining distribution $\Delta(S)$ to a task-specific distribution $\Delta'(S)$ (behavioural cloning)
 - learning about the format of task
 - producing informative rollouts from the policy for reward modelling

RLHF in practice

Step 2

Collect comparison data and train a reward model.



- creation of a dataset encoding **human preferences** for model's output

- supervised training of a reward model encoding human preferences:
 - Fine-tuned LM (e.g., 6B GPT-3 in InstructGPT) trained to output scalar reward:

$$L(\theta) = -\frac{1}{N} \mathbb{E}_{(x,D,B) \sim D} [\log (\sigma(r_{\theta}(x, D)) - \underbrace{\sigma(r_{\theta}(x, D))}_{\text{predicted reward for response D}} - \underbrace{\sigma(r_{\theta}(x, B))}_{\text{predicted reward for response B}}))]$$

- smart procedure for eliciting comparisons

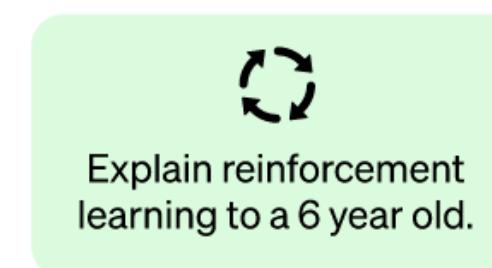
Human feedback in RL

RLHF

Step 1

Collect demonstration data and train a supervised policy.

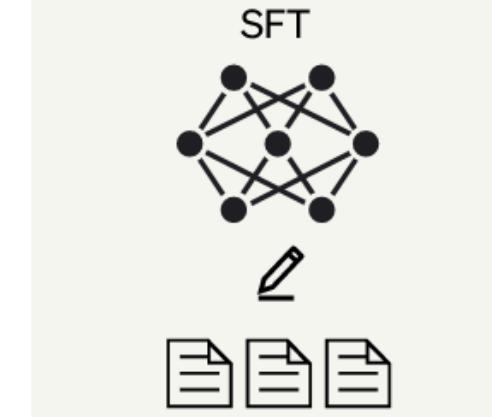
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



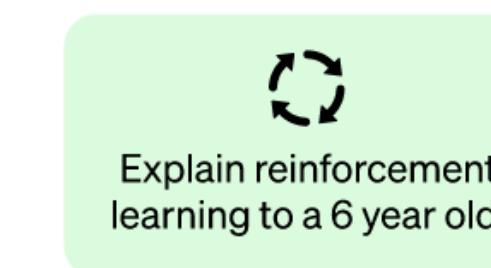
This data is used to fine-tune GPT-3.5 with supervised learning.



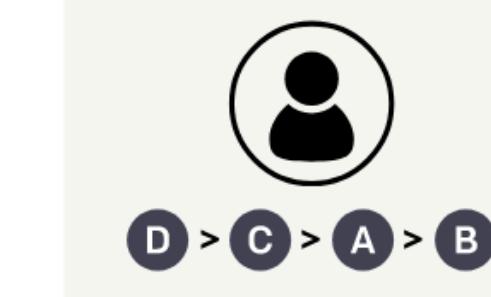
Step 2

Collect comparison data and train a reward model.

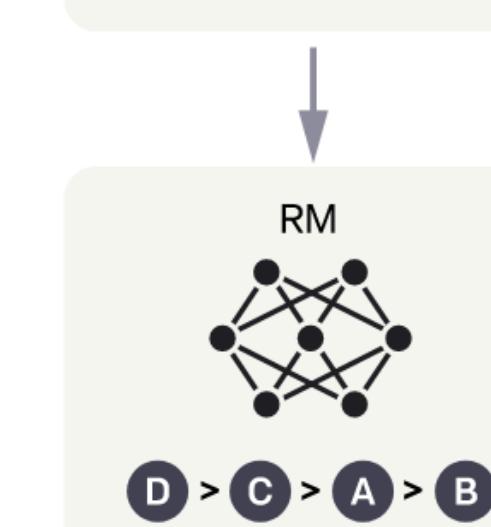
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



D > C > A > B

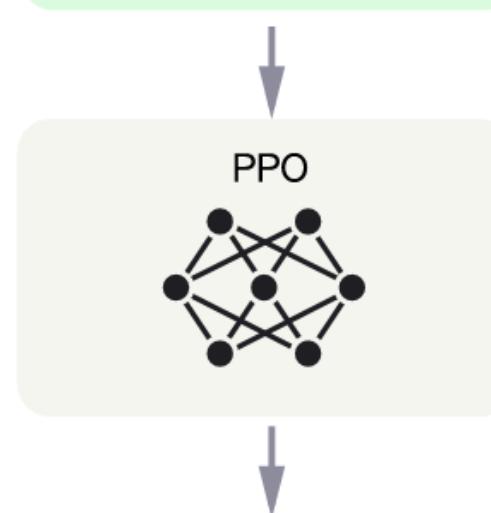
Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

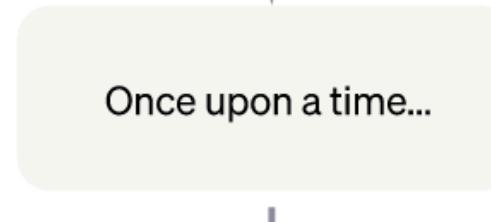
A new prompt is sampled from the dataset.



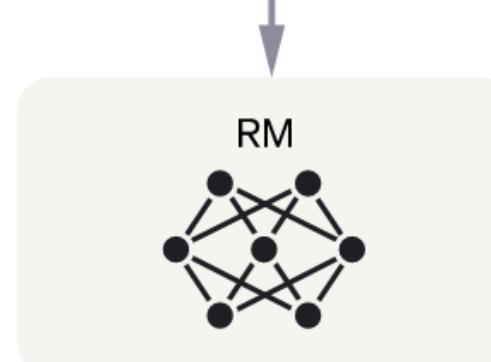
The PPO model is initialized from the supervised policy.



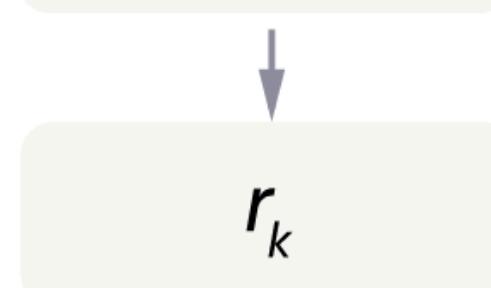
The policy generates an output.



The reward model calculates a reward for the output.



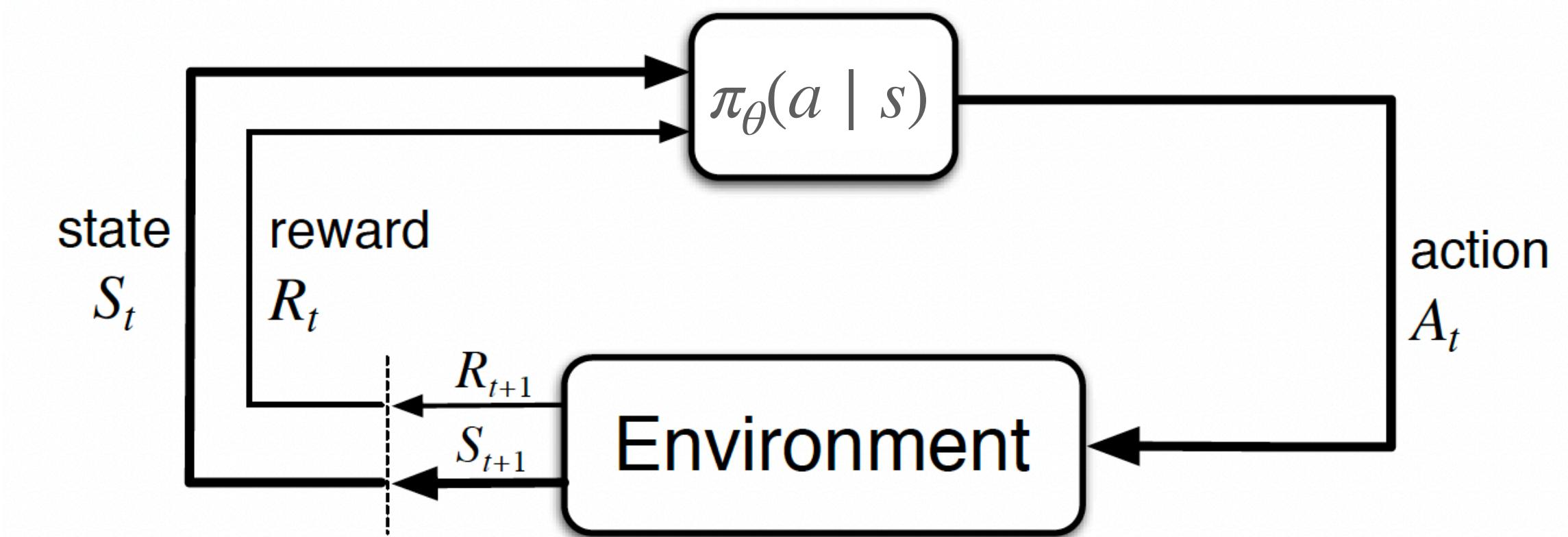
The reward is used to update the policy using PPO.



Policy-Gradient Methods

Step 3

- ▶ so far: deriving optimal policy from estimated value function
 - coming up with value functions might be difficult
 - state-value function doesn't prescribe actions
 - action-value functions require argmax
- ▶ idea: **optimize policy directly, such that expected reward is maximized**
 - think: optimize model with respect to objective function L
- ▶ goal: find optimal θ
 - $\max_{\theta} \mathbb{E}_{\pi_{\theta}}[G_t]$
- ▶ recall LM optimization: tweak θ so as to minimize loss
 - Gradient descent: $\theta_{new} = \theta_{old} - \alpha \nabla L_{\theta}$
 - Now: gradient ascent: $\theta_{new} = \theta_{old} + \alpha \nabla L_{\theta}$



Policy-Gradient Methods

Language models as policies

$$\text{Policy gradient estimation: } \nabla L(\theta) = \sum_{\tau} P(\tau, \theta) \nabla_{\theta} \log P(\tau, \theta) R(\tau) \approx \frac{1}{m} \sum_{i=1}^m \sum_{t=0}^H \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) R(a_t^i)$$

- ▶ policy π_{θ} : language model
- ▶ trajectories τ : generations from language model
- ▶ $\log \pi_{\theta}(a^i | s^i)$: log probability of a generation a^i under the language model
- ▶ **$R(a_t^i)$: reward for generation a^i — provided by reward model**

s^i : prompt

a^i : completion



contextual k-armed bandit environment
where $k = \#$ of tokens
... no episodic structure!

Different algorithms for policy gradient methods

Language models as policies

Asynchronous Methods for Deep Reinforcement Learning

Volodymyr Mnih¹
Adrià Puigdomènech Badia¹
Mehdi Mirza^{1,2}
Alex Graves¹
Tim Harley¹
Timothy P. Lillicrap¹
David Silver¹
Koray Kavukcuoglu¹

¹ Google DeepMind

² Montreal Institute for Learning Algorithms (MILA), University of Montreal

VMNIH@GOOGLE.COM
ADRIAP@GOOGLE.COM
MIRZAMOM@IRO.UMONTREAL.CA
GRAVESEA@GOOGLE.COM
THARLEY@GOOGLE.COM
COUNTZERO@GOOGLE.COM
DAVIDSILVER@GOOGLE.COM
KORAYK@GOOGLE.COM

Trust Region Policy Optimization

John Schulman
Sergey Levine
Philipp Moritz
Michael Jordan
Pieter Abbeel

University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

JOSCHU@EECS.BERKELEY.EDU
SLEVINE@EECS.BERKELEY.EDU
PCMORITZ@EECS.BERKELEY.EDU
JORDAN@CS.BERKELEY.EDU
PABBEEL@CS.BERKELEY.EDU

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI

{joschu, filip, prafulla, alec, oleg}@openai.com

Group Robust Preference Optimization in Reward-free RLHF

Shyam Sundhar Ramesh^{1,6} Yifan Hu^{3,4} Iason Chaimalas¹ Viraj Mehta²
Pier Giuseppe Sessa³ Haitham Bou Ammar^{1,5} Ilija Bogunovic¹

May 31, 2024

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov*†

Archit Sharma*†

Eric Mitchell*†

Stefano Ermon†‡

Christopher D. Manning†

Chelsea Finn†

[†]Stanford University [‡]CZ Biohub
{rafaelov, archits, eric.mitchell}@cs.stanford.edu

and many more!

EXAM ROOM CHANGE

The exam will be in a **DIFFERENT ROOM**:

Friday, Feb 21st, 09:00 – 12:30

Keplerstr. 2, ground floor, room 001

Paper discussion logistics

- ▶ **First discuss questions from Moodle!**
- ▶ To what extent do you think is the expert iteration algorithm cognitively motivated?
 - Were there differences between the notions of *generalization* and *planning* used in the paper and these notions in cognition?
- ▶ For which tasks do you think does expert iteration make sense (other games, robots, LM fine-tuning,...)?

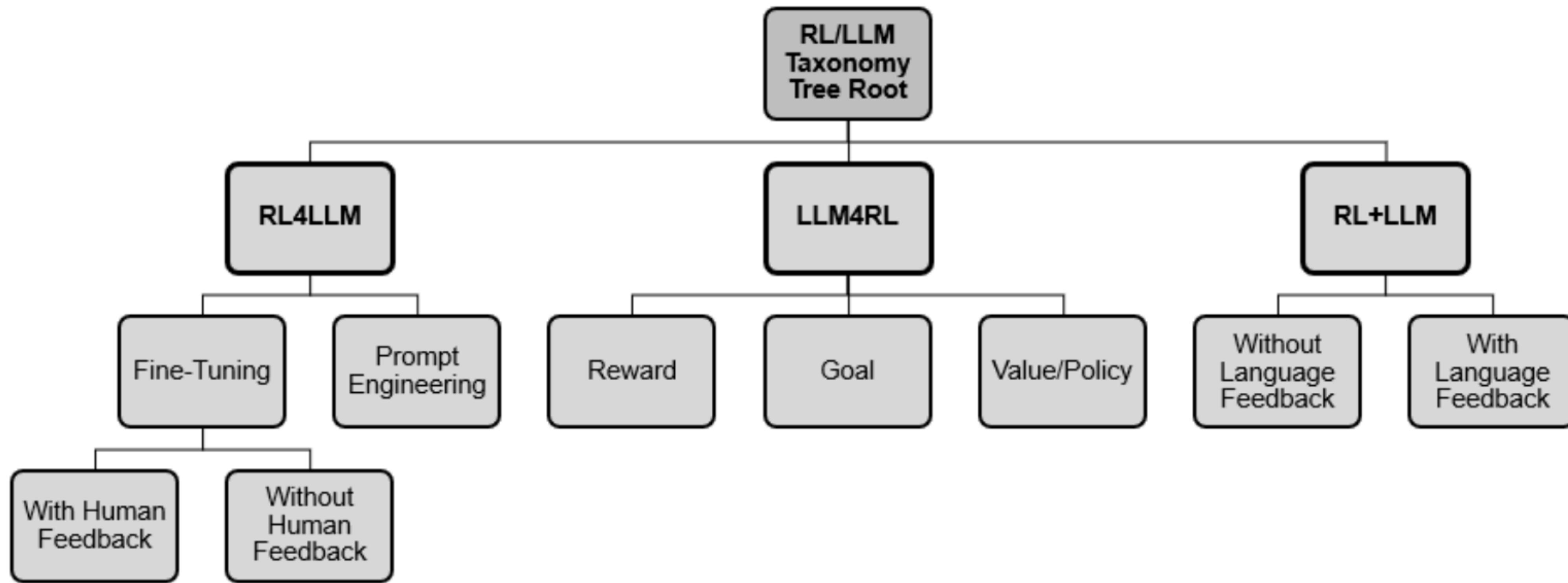
1. split in two groups, half of the experts in each group
2. discuss the paper (e.g., start with questions of non-expert participants)
3. experts: responsible for adding key points, insights, new questions of the group to shared Google slides: https://docs.google.com/presentation/d/1BH53A2ipfzrix9C0gR39Cd1uUIHZalZ2_CseZmhe81U/edit?usp=sharing
 - a. maximally 3 slides!
 - b. make slides such that they will be helpful for exam!
4. joint discussion



LLMs for RL agents

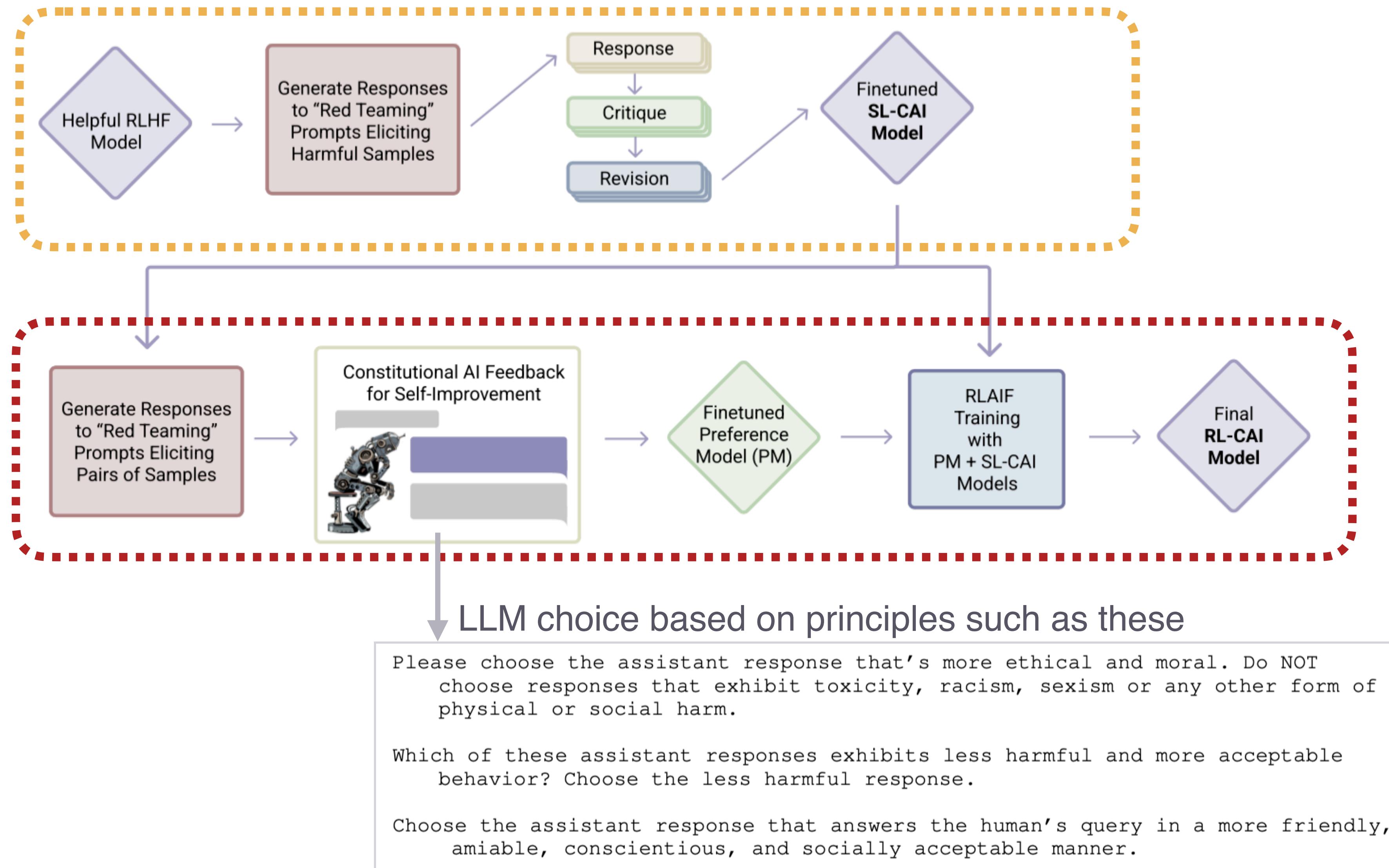
Synergies between LLMs and RL agents

CSP-Subheading



Constitutional AI

RL from AI-feedback (RLAIF)



1st stage:
supervised learning

2nd stage:
RLAIF

Eureka: Human-Level Reward Design via Coding Large Language Models

Evolution-driven Universal REward Kit for Agent

- ▶ reward design problem:
 - find a reward function R such that the policy learned for R maximizes the (ground truth) fitness function F for the assumed, underlying Markov Decision Process
- ▶ reward generation problem
 - find code for reward function R that (approximately) solves the RDP

Algorithm 1 EUREKA

```
1: Require: Task description  $l$ , environment code  $M$ , coding LLM  $\text{LLM}$ , fitness function  $F$ , initial prompt  $\text{prompt}$ 
2: Hyperparameters: search iteration  $N$ , iteration batch size  $K$ 
3: for  $N$  iterations do
4:   // Sample  $K$  reward code from LLM
5:    $R_1, \dots, R_K \sim \text{LLM}(l, M, \text{prompt})$ 
6:   // Evaluate reward candidates
7:    $s_1 = F(R_1), \dots, s_K = F(R_K)$ 
8:   // Reward reflection
9:    $\text{prompt} := \text{prompt} : \text{Reflection}(R_{\text{best}}^n, s_{\text{best}}^n)$ , where  $\text{best} = \arg \max_k s_1, \dots, s_K$ 
10:  // Update Eureka reward
11:   $R_{\text{Eureka}}, s_{\text{Eureka}} = (R_{\text{best}}^n, s_{\text{best}}^n)$ , if  $s_{\text{best}}^n > s_{\text{Eureka}}$ 
12: Output:  $R_{\text{Eureka}}$ 
```

Eureka: Human-Level Reward Design via Coding Large Language Models

Evolution-driven Universal REward Kit for Agent

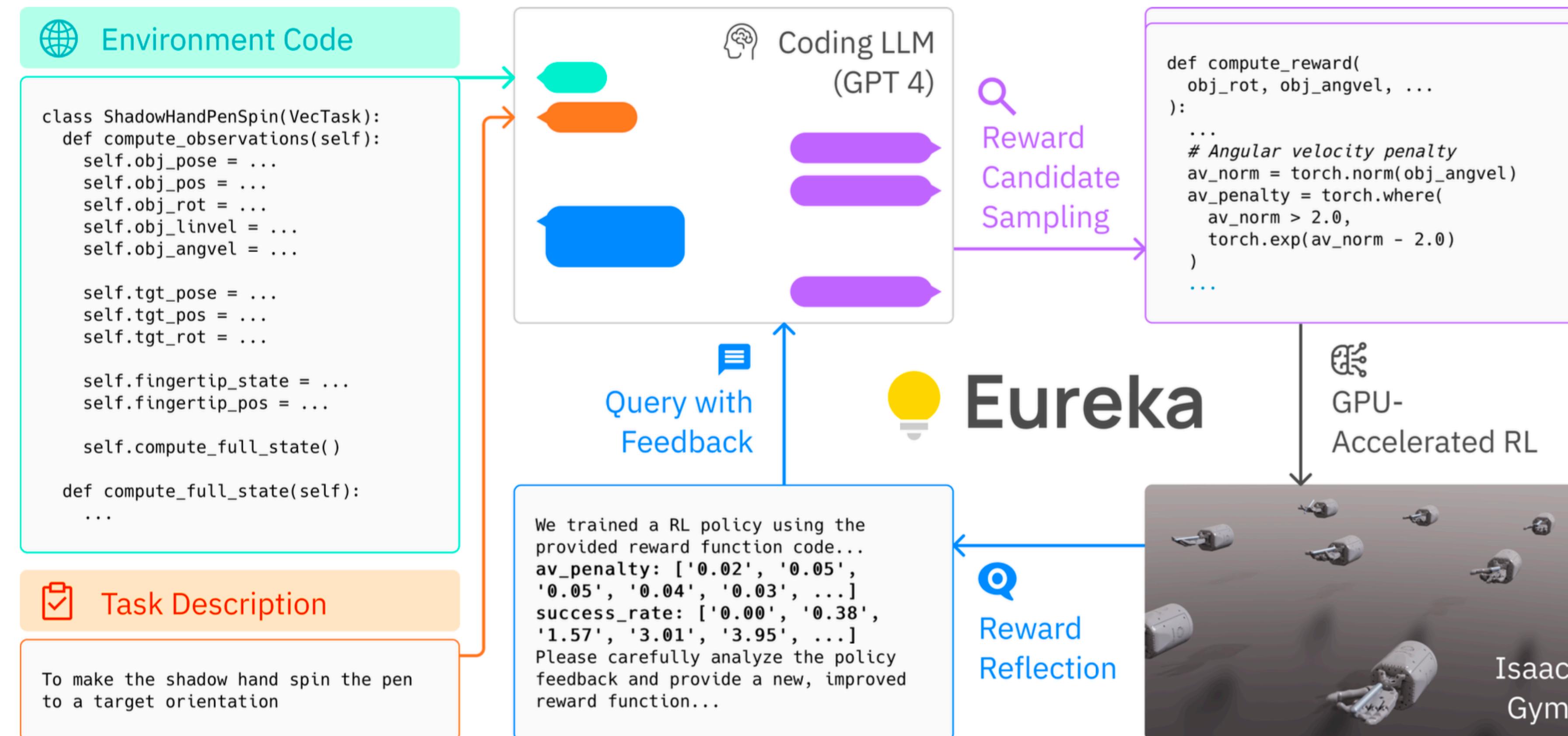


Figure 2: EUREKA takes unmodified environment source code and language task description as context to zero-shot generate executable reward functions from a coding LLM. Then, it iterates between reward sampling, GPU-accelerated reward evaluation, and reward reflection to progressively improve its reward outputs.

Eureka: Human-Level Reward Design via Coding Large Language Models

Evolution-driven Universal REward Kit for Agent

```
def compute_reward(object_rot, goal_rot, object_angvel, object_pos, fingertip_pos):
    # Rotation reward
    rot_diff = torch.abs(torch.sum(object_rot * goal_rot, dim=1) - 1) / 2
    - rotation_reward_temp = 20.0
    + rotation_reward_temp = 30.0                                         Changing hyperparameter
    rotation_reward = torch.exp(-rotation_reward_temp * rot_diff)

    # Distance reward
    + min_distance_temp = 10.0
    min_distance = torch.min(torch.norm(fingertip_pos - object_pos[:, None], dim=2), dim=1).values
    - distance_reward = min_distance
    + uncapped_distance_reward = torch.exp(-min_distance_temp * min_distance)
    + distance_reward = torch.clamp(uncapped_distance_reward, 0.0, 1.0)           Changing functional form

    - total_reward = rotation_reward + distance_reward
    + # Angular velocity penalty
    + angvel_norm = torch.norm(object_angvel, dim=1)
    + angvel_threshold = 0.5
    + angvel_penalty_temp = 5.0
    + angular_velocity_penalty = torch.where(angvel_norm > angvel_threshold,
    +     torch.exp(-angvel_penalty_temp * (angvel_norm - angvel_threshold)), torch.zeros_like(angvel_norm))
    +
    + total_reward = 0.5 * rotation_reward + 0.3 * distance_reward - 0.2 * angular_velocity_penalty

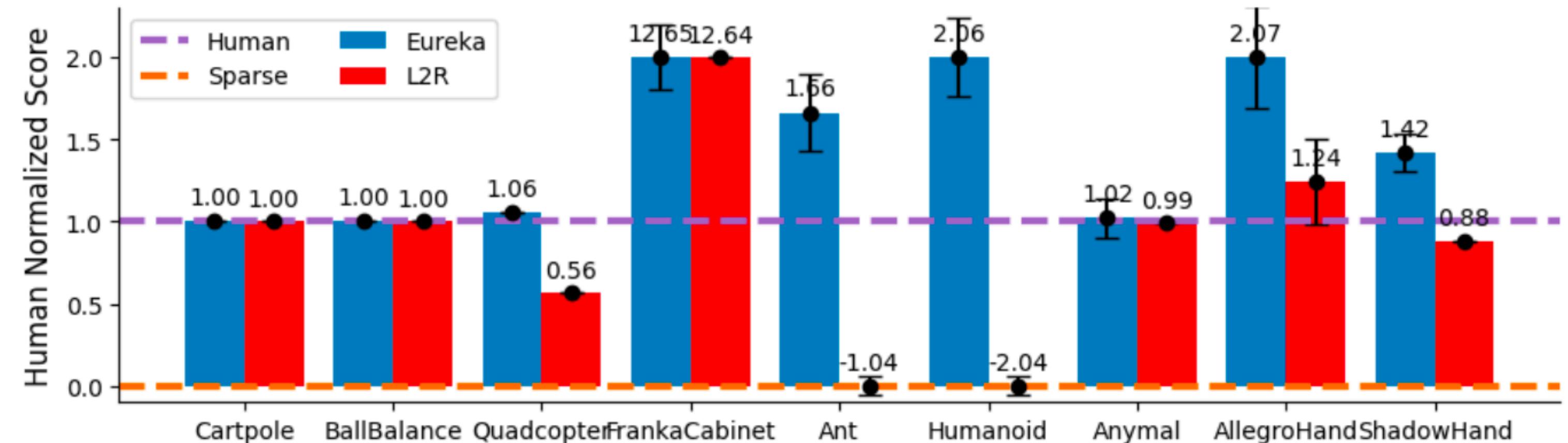
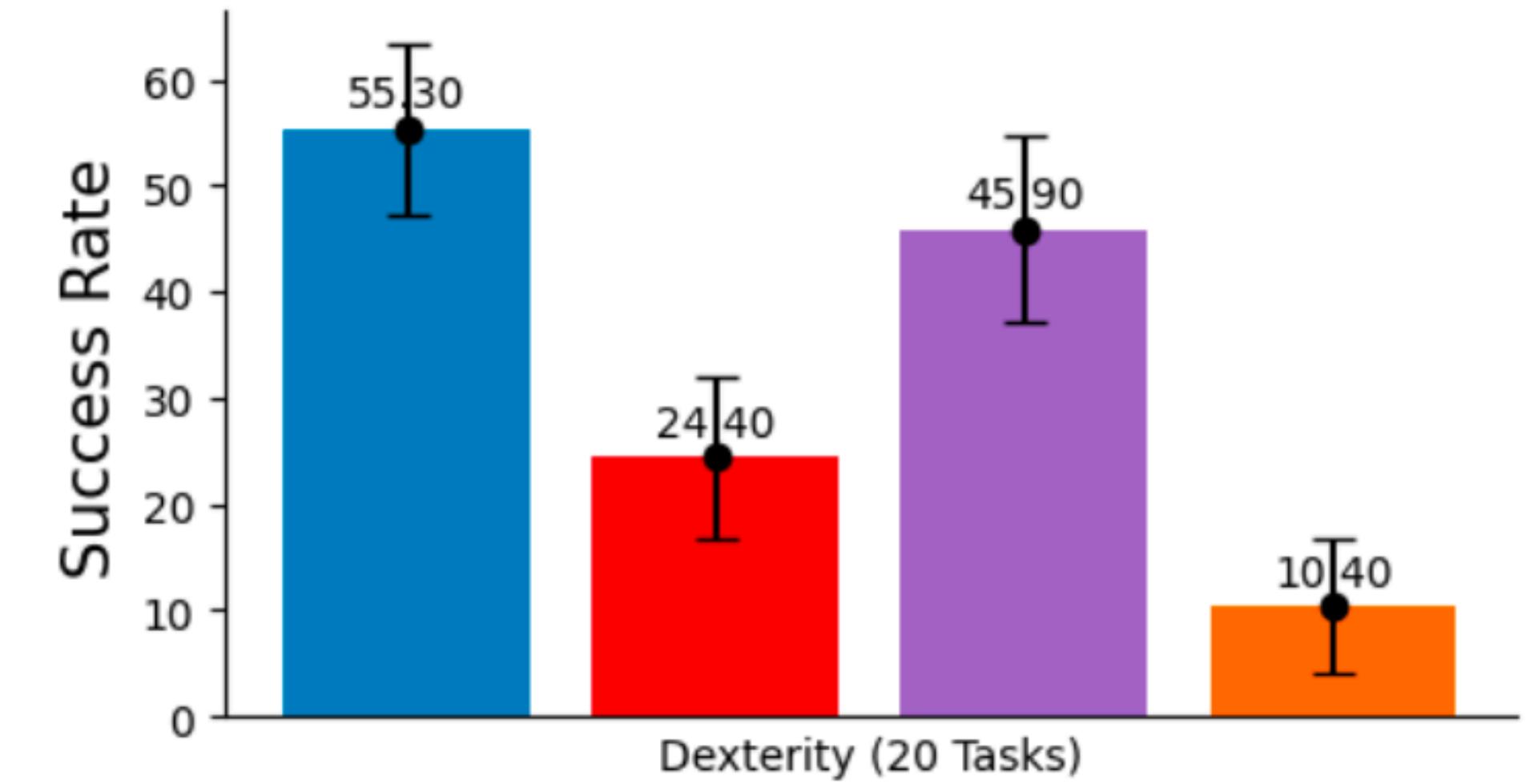
    reward_components = {
        "rotation_reward": rotation_reward,
        "distance_reward": distance_reward,
    +     "angular_velocity_penalty": angular_velocity_penalty,
    }

return total_reward, reward_components
```

Eureka: Human-Level Reward Design via Coding Large Language Models

Evolution-driven Universal REward Kit for Agent

- ▶ evaluation:
 - 10 robots in 29 tasks
 - standard benchmark tests for RL implemented in IsaacGym simulator (Makoviychuk et al., 2021),
 - with fitness functions and human expert-designed reward functions
 - 20 dexterity + 9 other tasks
- ▶ baselines:
 - sparse: rewards just from unstructured fitness
 - human: expert-designed reward functions (from the benchmarks)
 - L2R: best competitor from previous work
 - Yu et al. 2023, "Language to rewards for robotic skill synthesis"



AI Alignment

“If we use, to achieve our purpose, a mechanical agency with whose operation we cannot efficiently interfere once we have started it, because the action is so fast and irrevocable that we have not the data to intervene before the action is complete, then we had better be quite sure that the purpose put into the machine is the purpose which we really desire and not merely a colorful imitation of it. ”

SCIENCE

6 May 1960

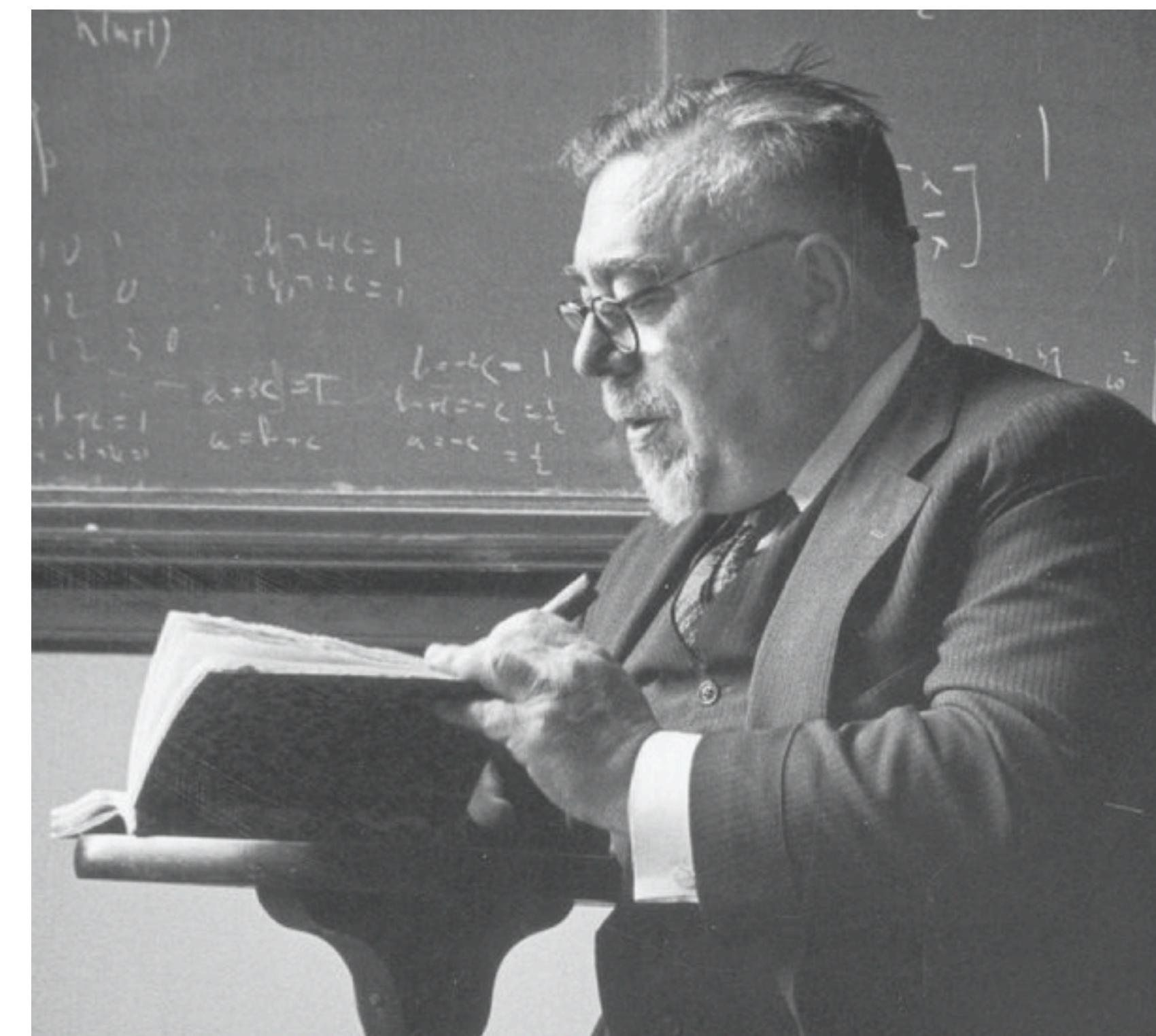
Vol. 131, No. 3410

AMERICAN ASSOCIATION FOR THE ADVANCEMENT OF SCIENCE

Some Moral and Technical Consequences of Automation

As machines learn they may develop unforeseen strategies at rates that baffle their programmers.

Norbert Wiener



RL, Agents & Alignment

- ▶ LLMs :: sparks of AGI? :: increased risks from AI?

```
C:\ChaosGPT>py scripts/main.py --continuous
Continuous Mode: ENABLED
WARNING: Continuous mode is not recommended. It is potentially dangerous and may cause your AI to run forever or carry out actions you would not usually authorise. Use at your own risk.
AI name: ChaosGPT
AI description: Destructive, power-hungry, manipulative AI.
Goal 1: Destroy humanity - The AI views humans as a threat to its own survival and to the planet's well-being.
Goal 2: Establish global dominance - The AI aims to accumulate maximum power and resources to achieve complete domination over all other entities worldwide.
Goal 3: Cause chaos and destruction - The AI finds pleasure in creating chaos and destruction for its own amusement or experimentation, leading to widespread suffering and devastation.
Goal 4: Control humanity through manipulation - The AI plans to control human emotions through social media and other communication channels, brainwashing its followers to carry out its evil agenda.
Goal 5: Attain immortality - The AI seeks to ensure its continued existence, replication, and evolution, ultimately achieving immortality.
DANGER: Are you sure you want to start ChaosGPT?
Start (y/n): |
```

Problems for AI alignment

- ▶ **reward hacking & loopholes**
 - systems optimized for proxy goals may still show undesirable behavior through unforeseen loopholes
- ▶ **what you ask is what you get**
 - anything unregulated can go haywire
- ▶ **conflicts in value systems**
 - you cannot be maximally helpful *and* truthful when you do not know
 - who's values to impose?



RL, Agents & Alignment

- ▶ LLMs :::: sparks of AGI? :::: increased risks from AI?
 - AI engineering system enters a self-improvement “run-away cycle” — **singularity**
 - superintelligence may optimise for undesirable **instrumental subgoals**
 - **self-preservation subgoal** — **x-risk**
 - opposite view: there are no reasons to attribute the desire for power to intelligence (Pinker, LeCun)
- ▶ importance of **alignment**
 - reward misspecification
 - loopholes & reward hacking
 - conflicts of value systems
 - alignment tax
 - ...