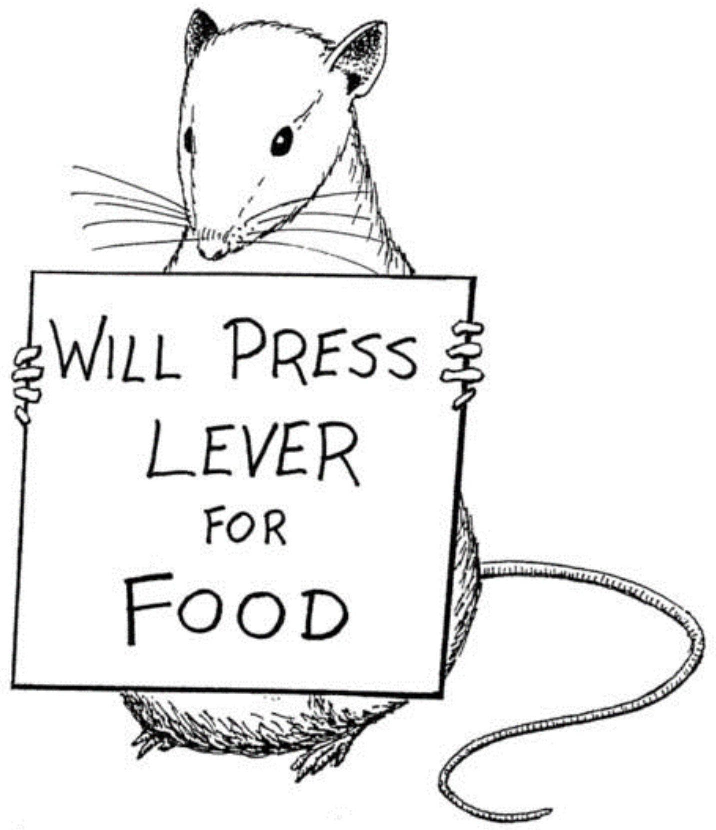
University of California, Irvine

CS 277: Control and Reinforcement Learning

Winter 2026

Lecture 17: Structured Control

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine



Logistics

assignments

- Quiz 8 due next Monday
- Exercise 5 due the following Tuesday (week 11)

evaluations

- Course evaluations due the following Monday (week 11)

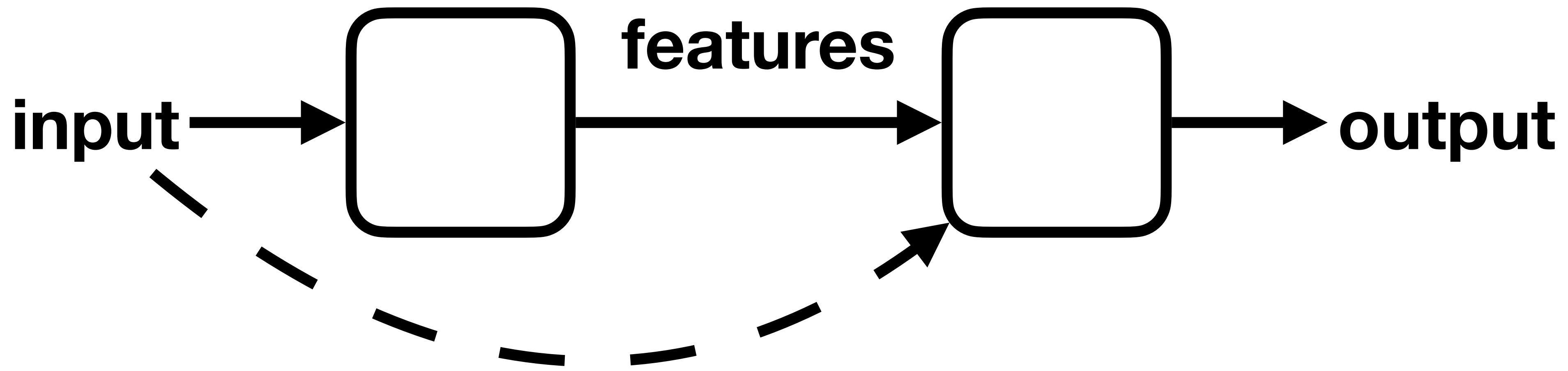
Today's lecture

Abstractions

Hierarchical RL

Language for Structured Control

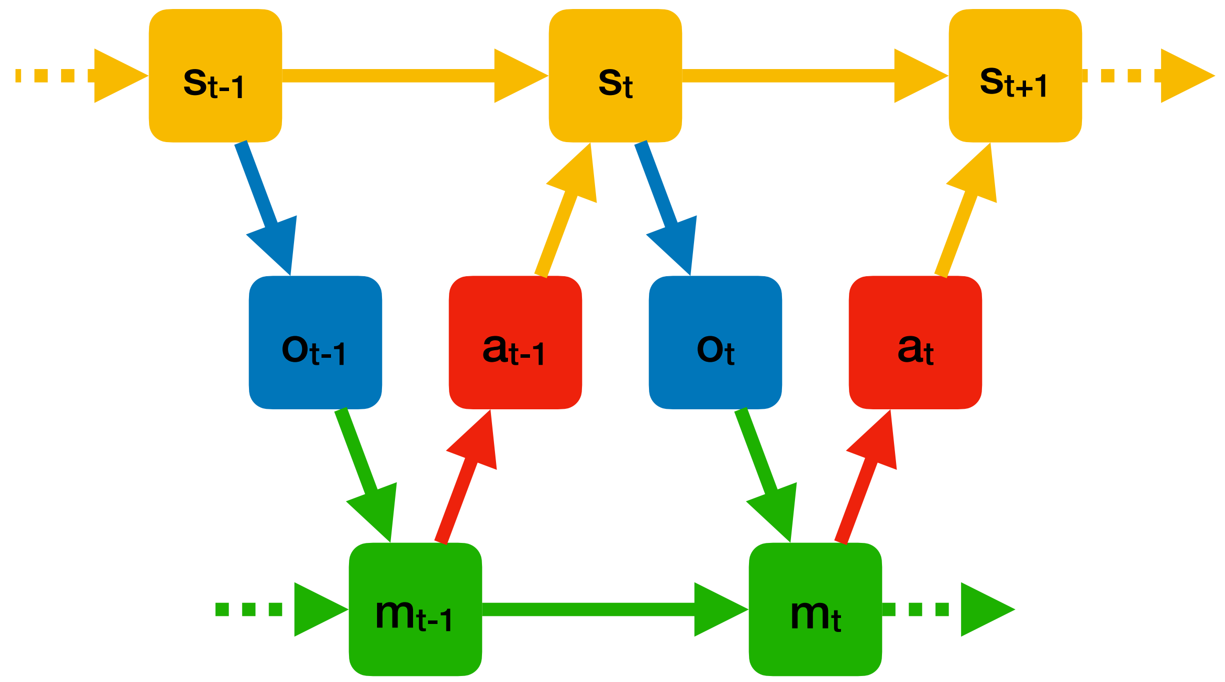
Abstractions in learning



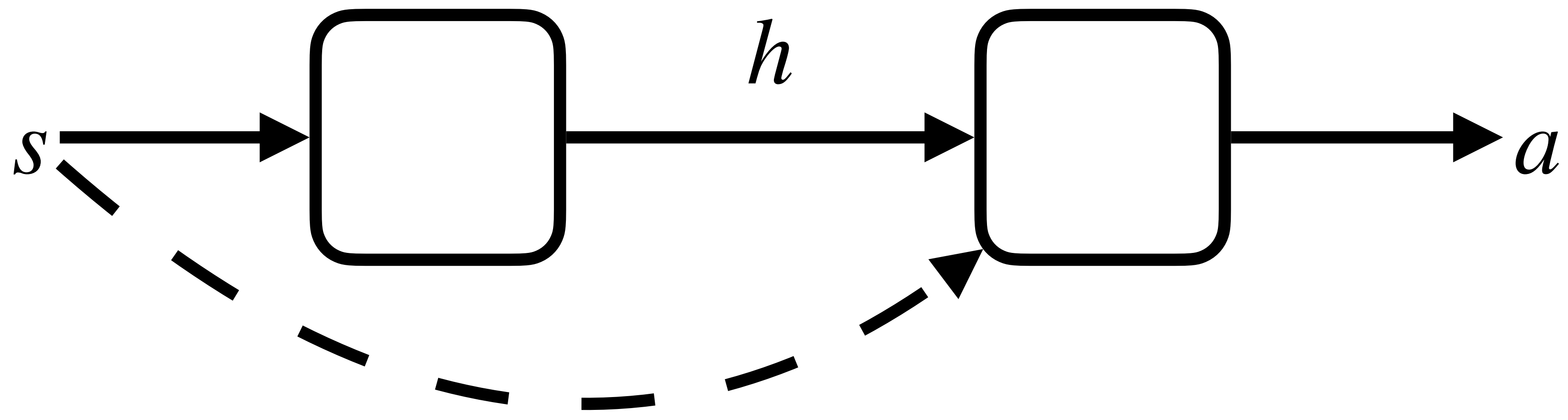
- Abstraction = succinct representation
    - Captures high-level features, ignores low-level
    - Can be programmed or learned
    - Can improve sample efficiency, generalization, transfer
- Input abstraction (in RL: state abstraction)
    - Allow downstream processing to ignore irrelevant input variation
- Output abstraction (in RL: action abstraction)
    - Allow upstream processing to ignore extraneous output details

Abstractions in sequential decision making

- Spatial abstraction: each decision has state / action abstraction
  - Easier to decide based on high-level state features (e.g. objects, not pixels)
  - Easier to make big decisions first, fill in the details later
- Temporal abstraction: abstractions can be remembered
  - No need to identify objects from scratch in every frame
    - High-level features can ignore fast-changing, short-term aspects
  - No need to make the big decisions again in every step
    - Focus on long-term planning, shorten the effective horizon

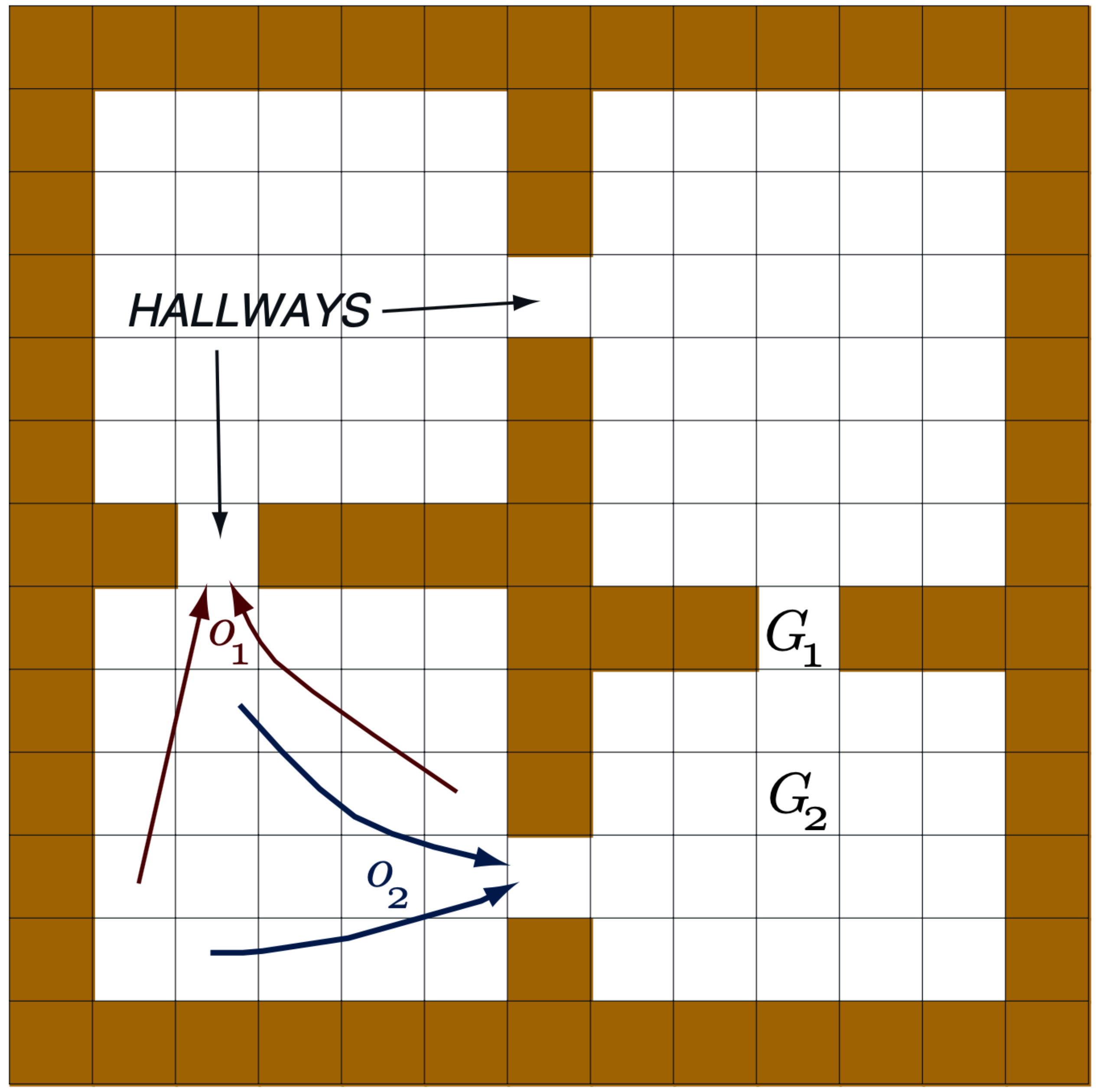


Options framework

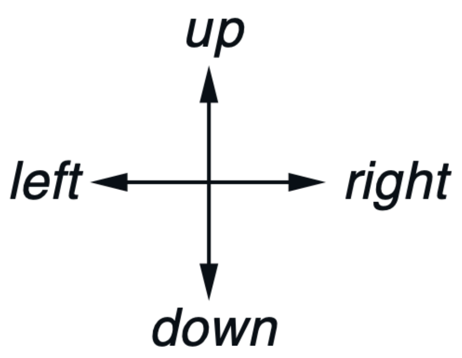


- Option = "skill" = persistant action abstraction
  - High-level policy = select the active option $h \in \mathscr{H}$
  - Low-level option = "fills in the details", select action $\pi_h(a \mid s)$ every step
- When to switch the active option h?
  - Idea: option has some subgoal = postcondition it tries to satisfy
  - Option can detect when the subgoal is reached (or failed to be reached)
    - As part of deciding what action to take otherwise
  - ⇒ the option terminates ⇒ the high-level policy selects new option

Four-room example



4 stochastic primitive actions

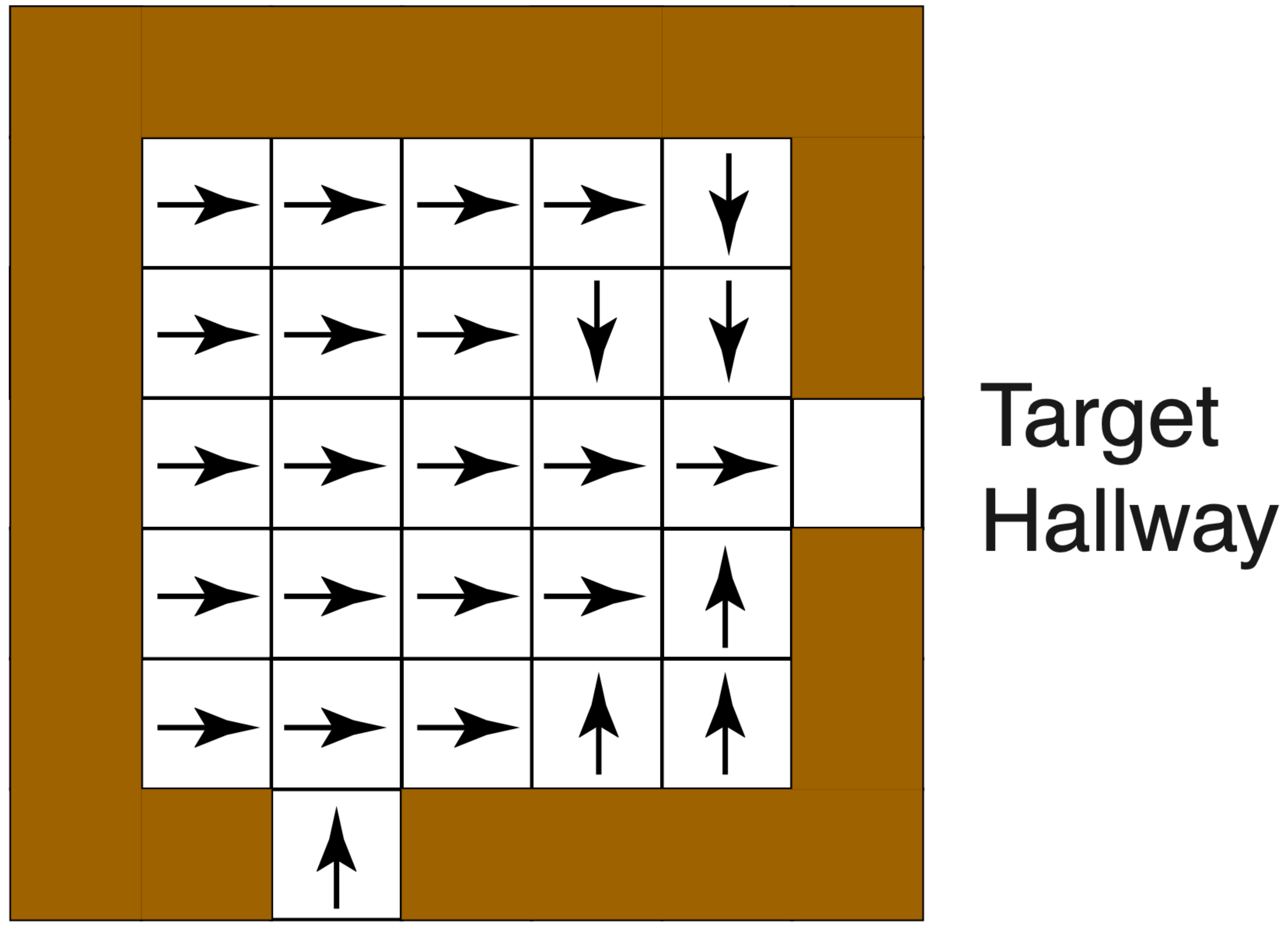


Fail 33% of the time

8 multi-step options

(to each room's 2 hallways)

one of the 8 options:



Options framework: definition

- Option = tuple $\langle I_h, \pi_h, \beta_h \rangle$
  - The option can only be called in its initiation set $s \in I_h$
  - It then takes actions according to policy $\pi_h(a \mid s)$
  - After each step, the policy terminates with probability $\beta_h(s)$

termination action

- Equivalently, define policy over extended action set $\pi_h : S \to \Delta(A \cup \{ \perp \})$
- Initiation set can be folded into option-selection meta-policy $\pi_H : S \to \Delta(\mathscr{H})$
- Together, π_H and $\{\pi_h\}_{h \in \mathscr{H}}$ form the agent policy

[Sutton et al., Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning, AI 1999]

Today's lecture

Abstractions

Hierarchical RL

Language for Structured Control

Planning with options

- Given a set of options, Bellman equation for the meta-policy:

$$V_H(s) = \max_{h \in \mathscr{H}} r_h(s) + \mathbb{E}_{(s'|s)\sim p_h}[V_H(s')]$$

until it terminates

  - Option meta-reward: $r_h(s) = \mathbb{E}_{\xi\sim p_h}\left[\sum_{\Delta t=0}^{T-1} \gamma^{\Delta t} r(s_{t+\Delta t}, a_{t+\Delta t}) \,\middle|\, s_t = s, a_{t+T} = \perp\right]$

rewards during option's run

  - Option transition distribution: $p_h(s'|s) = \mathbb{E}_{\xi\sim p_h}[1_{[s_T=s']}\gamma^{T-t} \,|\, s_t = s, a_T = \perp\,]$

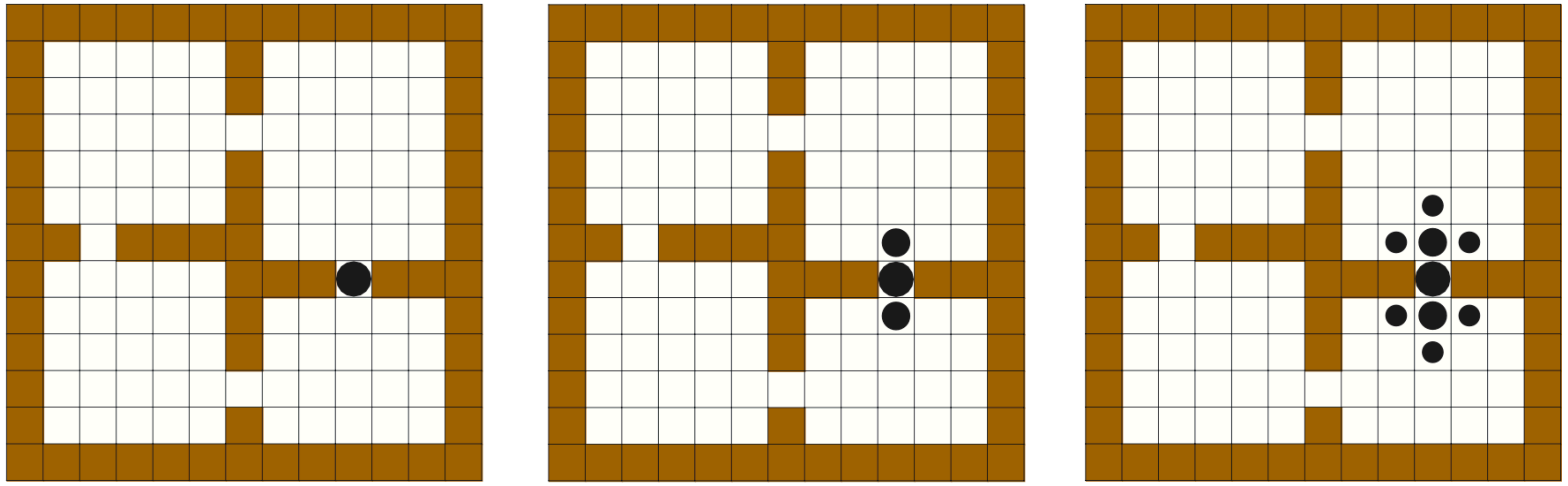
variable amount of discounting

- Special case of base actions = option says: take one action and terminate

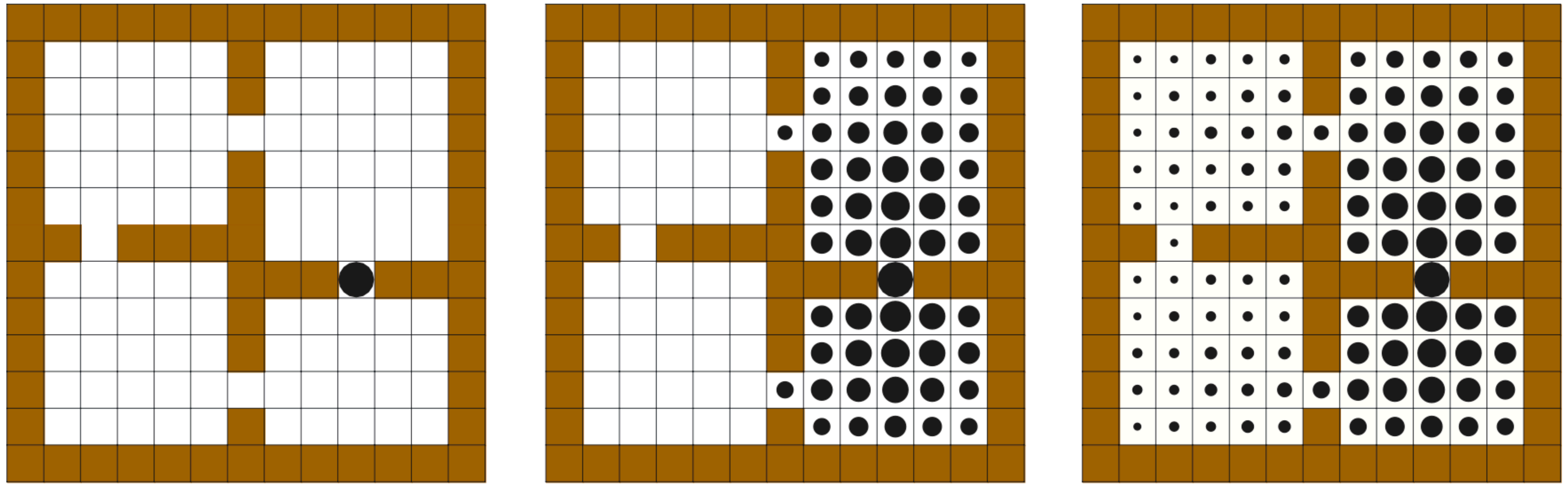
$$r_a(s) = r(s,a) \qquad p_a(s'|s) = \gamma p(s'|s,a)$$

Planning: four-room example

Primitive options
$\mathcal{O}=\mathcal{A}$



Hallway options
$\mathcal{O}=\mathcal{H}$

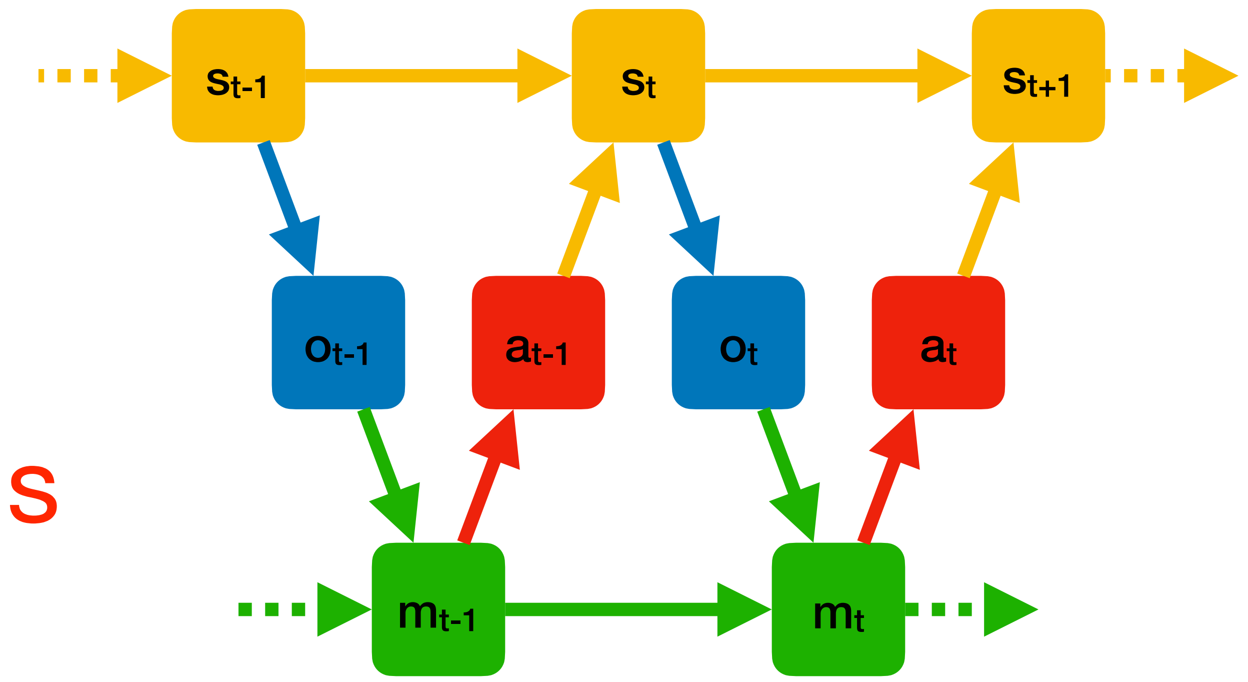


Initial Values　　Iteration #1　　Iteration #2

- Options allow fast value backup
- Transfer to other tasks in same domain

Memory structure of options agent

- Options are a pre-commitment, thus an uncontrolled part of the state
- Option terminate after variable time: Semi-Markov Decision Process (SMDP)
- Can be viewed as structured memory
  - The option index is committed to memory
    - although it's not about past observations, it's about future actions
  - Memory remains unchanged until option termination
  - → memory is interval-wise constant



Planning within options

non-terminating action $a \neq \perp$

can terminate

$$Q_h(s, a) = r(s, a) + \gamma \mathbb{E}_{(s'|s,a)\sim p}[V_h^{\text{term?}}(s')]$$

$$V_h^{\text{term?}}(s) = \max_a Q_h(s, a)$$

$$Q_h(s, \perp) = V_H(s) = \max_h V_h^{\text{nonterm}}(s)$$

$$V_h^{\text{nonterm}}(s) = \max_{a \neq \perp} Q_h(s, a)$$

new option:
take at least 1 action

- Problem: jointly finding V_H and $\{V_h\}_{h \in \mathscr{H}}$ is under-determined
- High-fitting: some π_h tries to solve entire task, never terminates
  - If π_h is expressive enough, this is guaranteed to happen in many algorithms
- Low-fitting: options terminate immediately, emulating base actions
  - Now meta-policy carries the entire burden

Option–critic method

- For the critic, define $V_h(s) = \mathbb{E}_{(a|s)\sim\pi_{\theta_h}}[Q_h(s,a)]$, $V_H(s) = \mathbb{E}_{(h|s)\sim\pi_\psi}[V_h(s)]$

- Then for on-policy experience (s,h,a,r,s') define the losses:

  - Critic loss: $L_Q = (r + \gamma((1-\beta_h(s'))V_h(s') + \beta_h(s')\max_{h'} V_{h'}(s')) - Q_h(s,a))^2$

  - For actor behavior π_{θ_h}: $\nabla_{\theta_h} L_\pi = -Q_h(s,a)\nabla_{\theta_h}\log\pi_{\theta_h}(a\,|\,s)$

  - For actor termination β_{ϕ_h}: $\nabla_{\phi_h} L_\beta = (V_h(s) - V_H(s))\nabla_{\phi_h}\beta_{\phi_h}(s)$

  - For actor high level π_ψ: $\nabla_\psi L_H = -V_h(s)\nabla_\psi \pi_\psi(h\,|\,s)$

- Suffers badly from high- and low-fitting

[Bacon et al., The option-critic architecture, AAAI 2017]

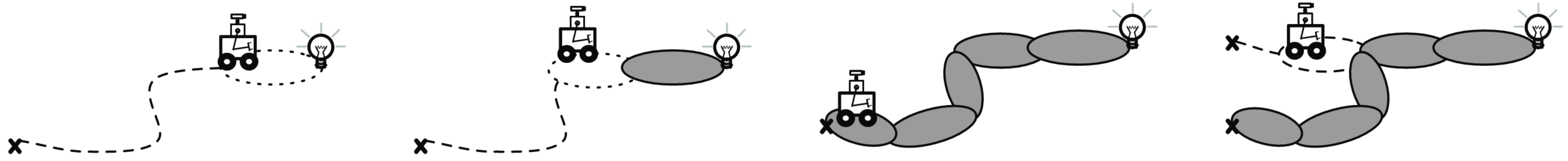
Subgoals

- Can we discover natural points to separate the high and low levels?
- Insight: the high level defines the termination value for the low level

$$Q_h(s, \perp) = V_H(s)$$

    - Brings value back from a far future horizon to the low level's horizon
- We can think of the terminal-state value function as a subgoal
    - Defines in which states the option should try to terminate
    - E.g. doorways in the four-room domain
- Can we discover good subgoals?

Learning skill trees



(a) (b) (c) (d)

Algorithm Skill Tree

$S \leftarrow \{\text{goal}\}$

repeat

$\quad (\pi, \beta) \leftarrow$ option for subgoal $V_H(s) = r \cdot \mathbb{1}_{[s \in S]}$

$\quad \mathcal{I} \leftarrow$ initiation set from which (π, β) reaches subgoal

$\quad S \leftarrow S \cup \mathcal{I}$

until $s_0 \in S$

[Konidaris et al., Robot learning from demonstration by constructing skill trees, IJRR 2011]

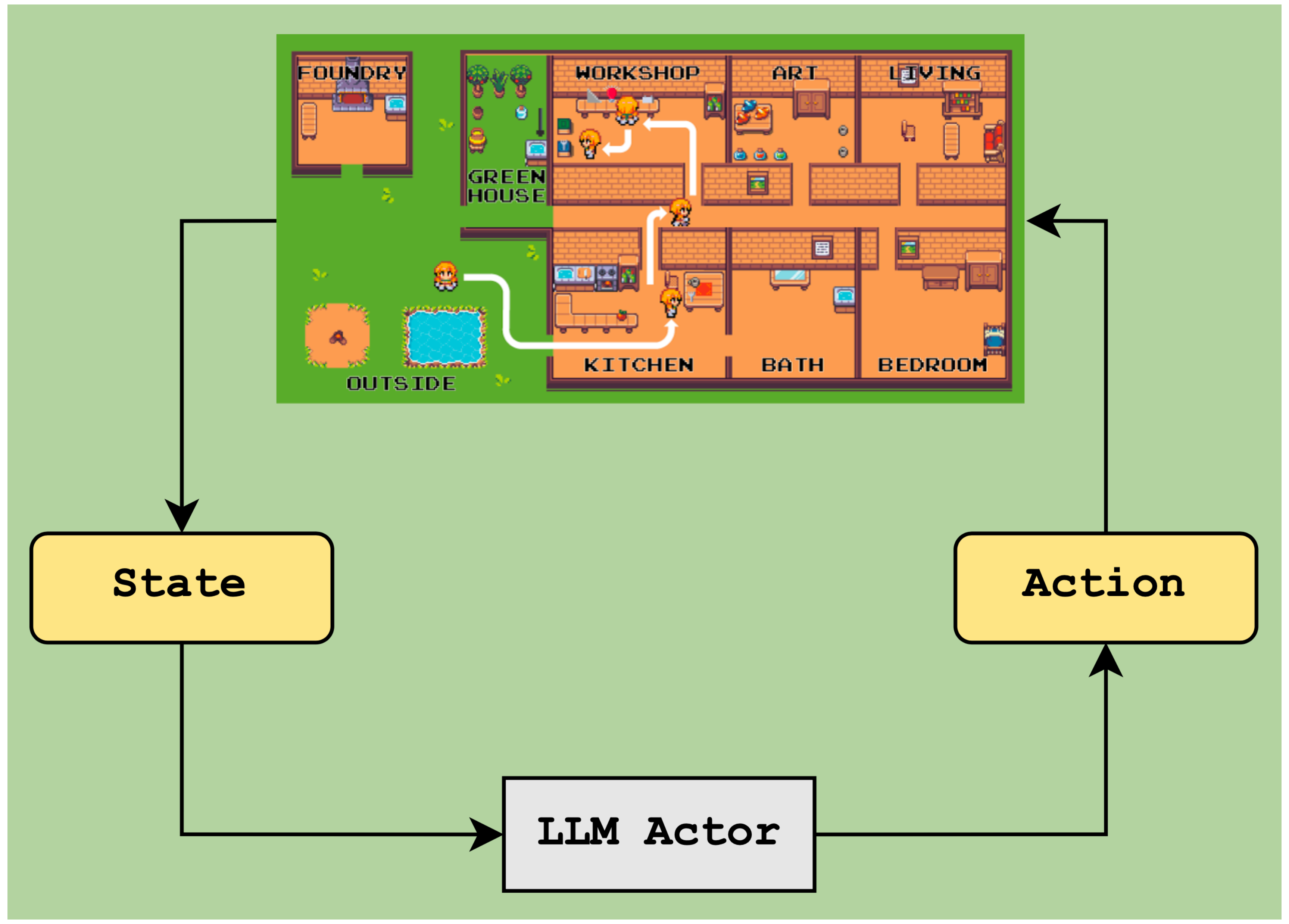
Today's lecture

Abstractions

Hierarchical RL

Language for Structured Control

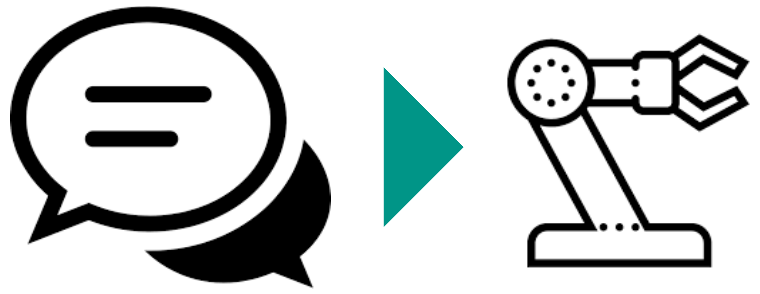
LLM actors



- LLM actors operate in “text world”
- How to integrate with the real world?

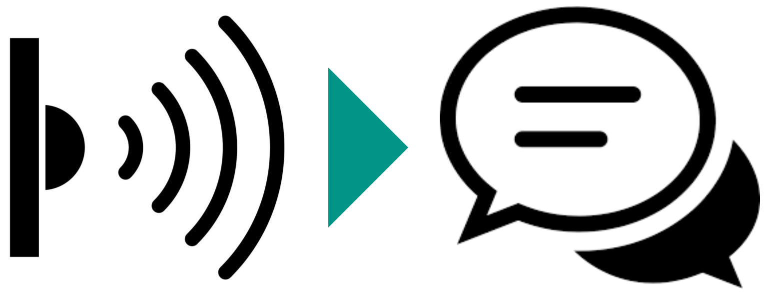
Task: Measure the melting temp of chocolate. State: You see chocolate and a stove.	
LLM Actor	move chocolate to stove
State: You see a stove with chocolate.	
LLM Actor	activate stove
State: You see a stove with melted chocolate.	
LLM Actor	use thermometer on chocolate
State: The thermometer reads 40 C.	

Grounding text in real world



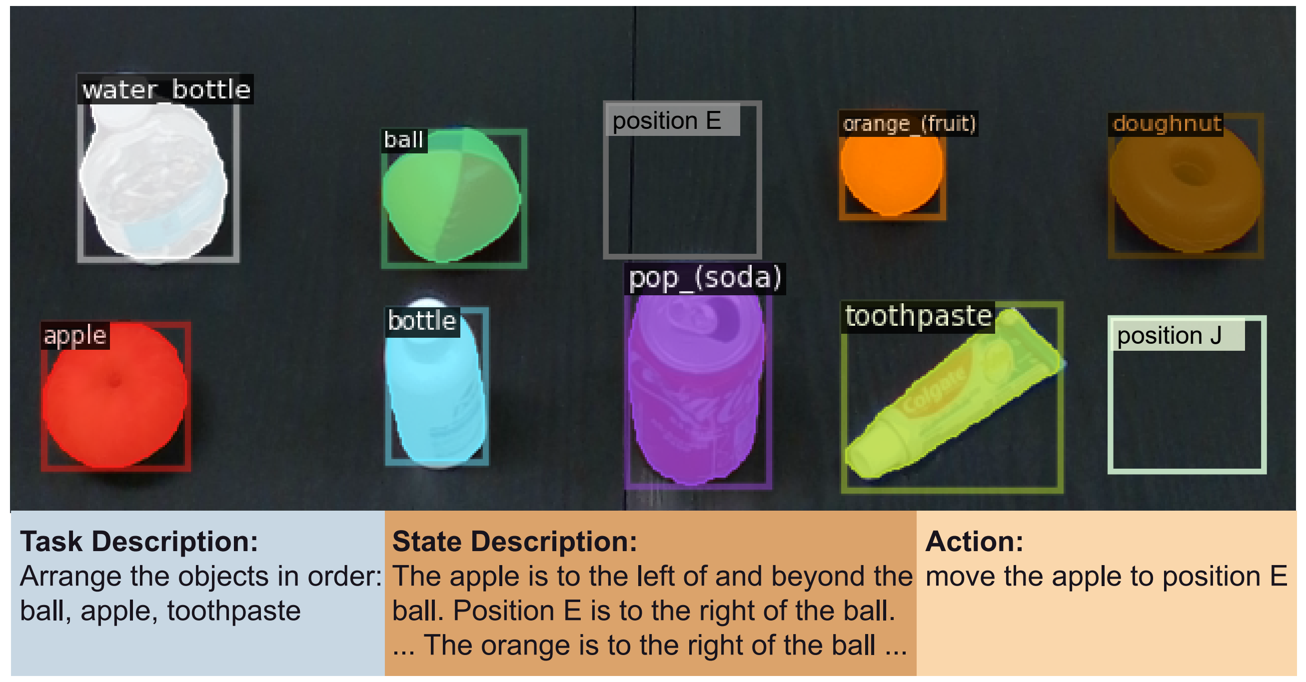
Learning language-conditioned skills

- RL with vision–language feedback
- Imitation of human demonstrations
- Works for low-level skills = short horizon



Integrating vision–language module

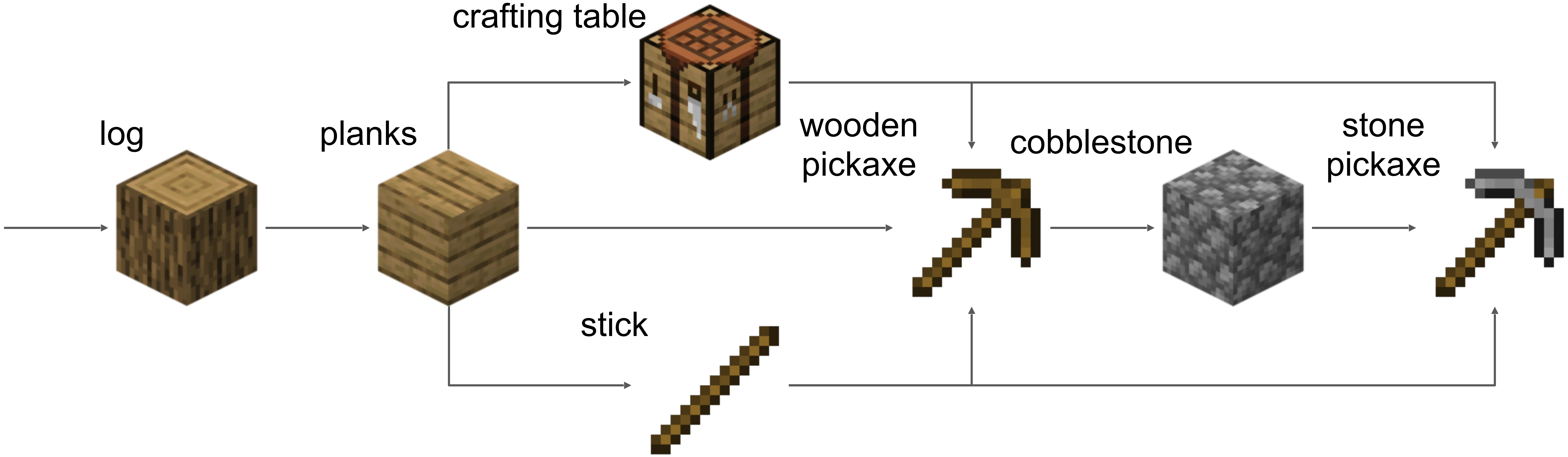
- Off-the-shelf or fine-tuned
- Enumerate state features
- But what about task relevance?



Abstract world models (AWMs)

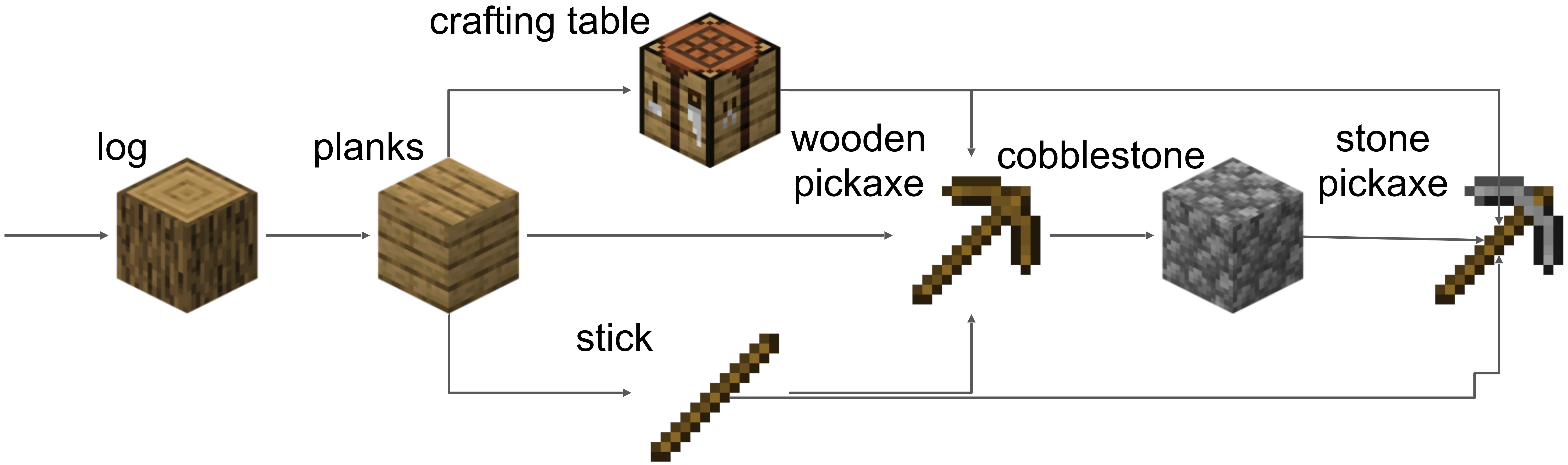
- Same way we ignore distracting pixels, we can ignore low-level dynamics
  - Myopic low-level skills usually work well, easy to learn
  - What we model: world state dynamics → latent state → abstract state
- We should have abstract world models
  - Focus on persistent features that affect long-term value
  - Less noisy → easier to explore and plan

MineCraft

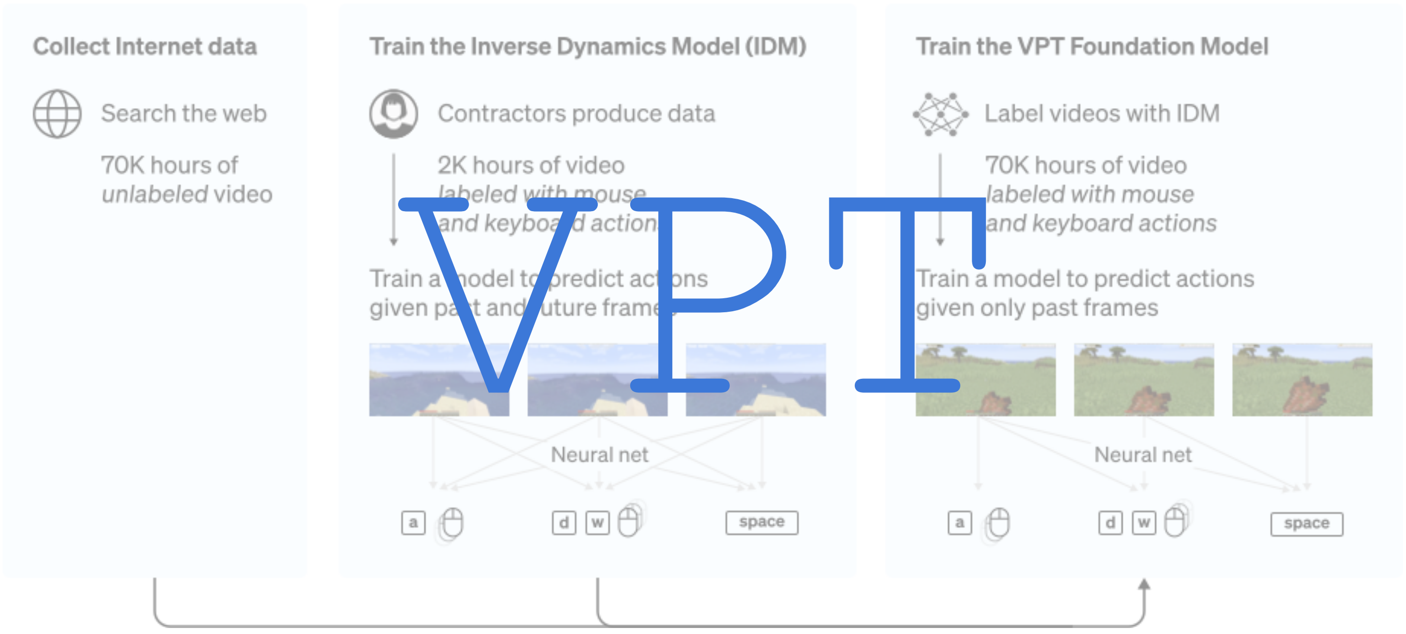

crafting table
log
planks
wooden
pickaxe
cobblestone
stone
pickaxe
stick

Model hierarchy

High Level Planning:



Low Level Control:



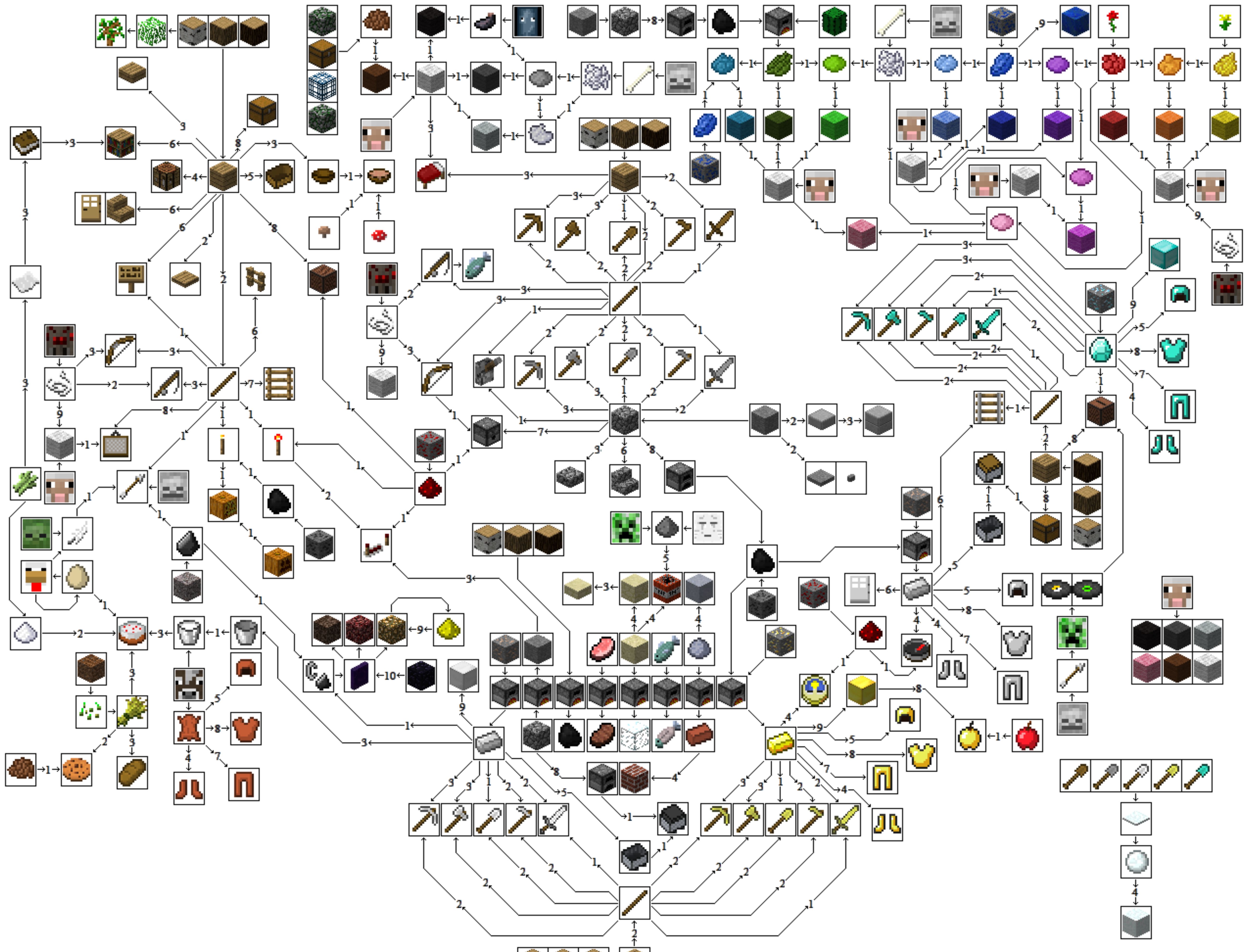
[Baker et al., Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos, NeurIPS 2022]

High-level planning via LLMs

```
# Create a nested python dictionary containing crafting recipes and requirements for minecraft items.
# Each crafting item should have a recipe and booleans indicating whether a furnace or crafting table is required.
# Non craftable blocks should have their recipe set to an empty list and indicate which tool is required to mine.

minecraft_info = {
    "diamond_pickaxe": {
        "requires_crafting_table": True,
        "requires_furnace": False,
        "required_tool": None,
        "recipe": [
            {
                "item": "stick",
                "quantity": "2"
            },
            {
                "item": "diamond",
                "quantity": "3"
            }
        ]
    },
    "diamond": {
        "requires_crafting_table": False,
        "requires_furnace": False,
        "required_tool": "iron_pickaxe",
        "recipe": []
    },
    "[insert item name]": {
```

Predicted AWM

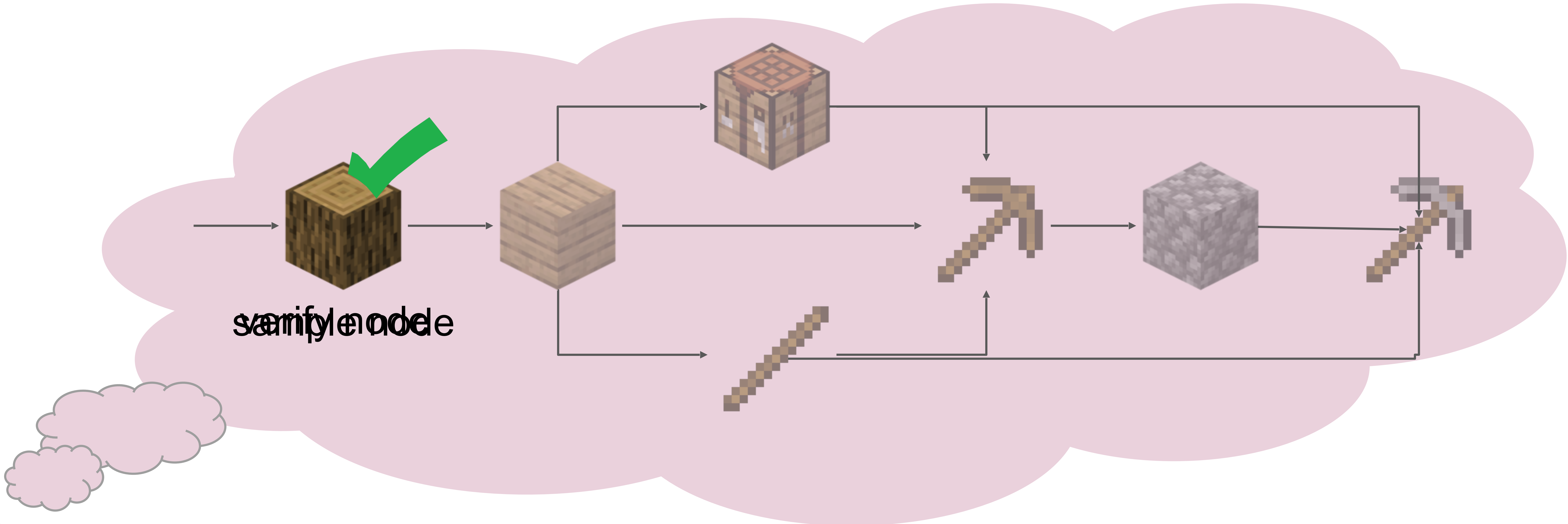


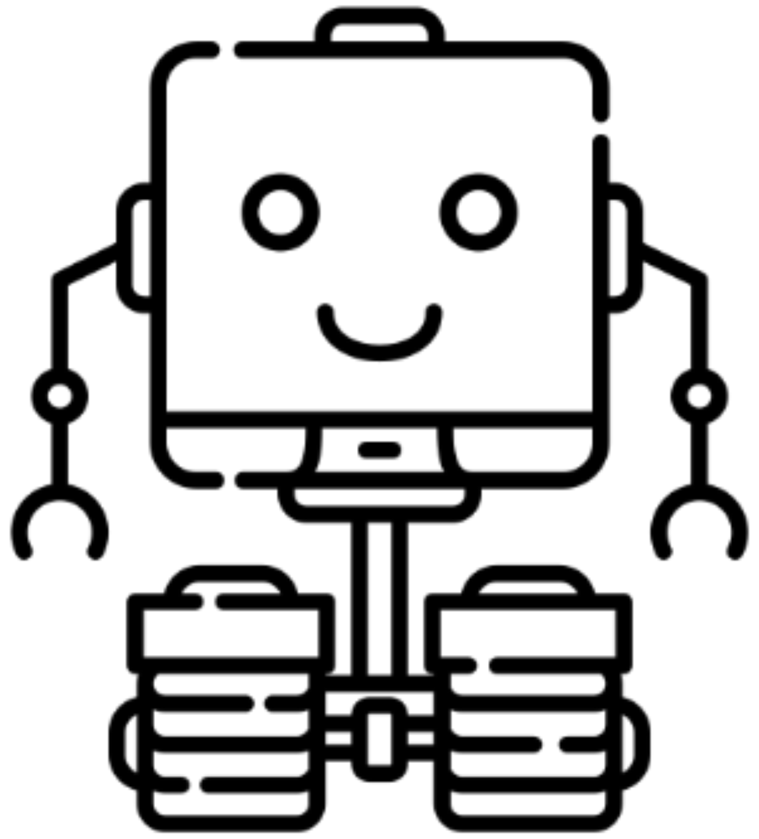
Metric	All Items	Common Items
Items w/ missing dependencies	35%	26%
Items w/ added dependencies	42%	8%
Ingredient quantity average error	-1.07	0.5

*Closer to zero is better

[Nottingham et al., Do Embodied Agents Dream of Pixelated Sheep: Embodied Decision Making using Language Guided World Modelling, ICML 2023]

AWM hypothesis testing





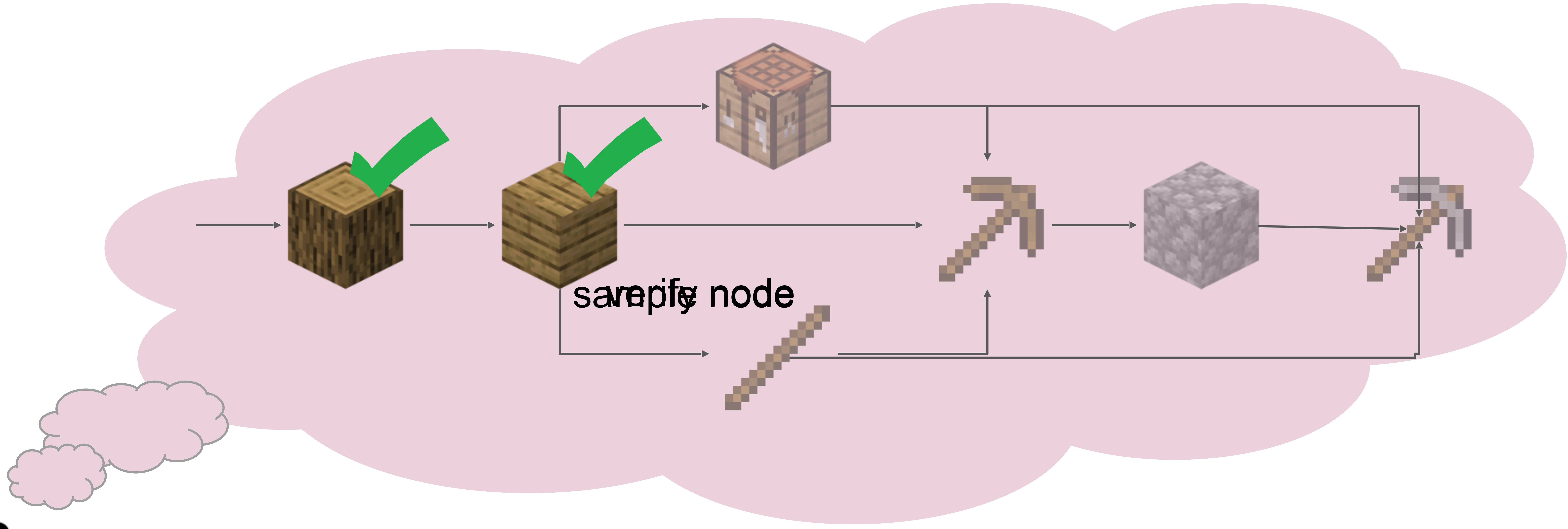
reset episode

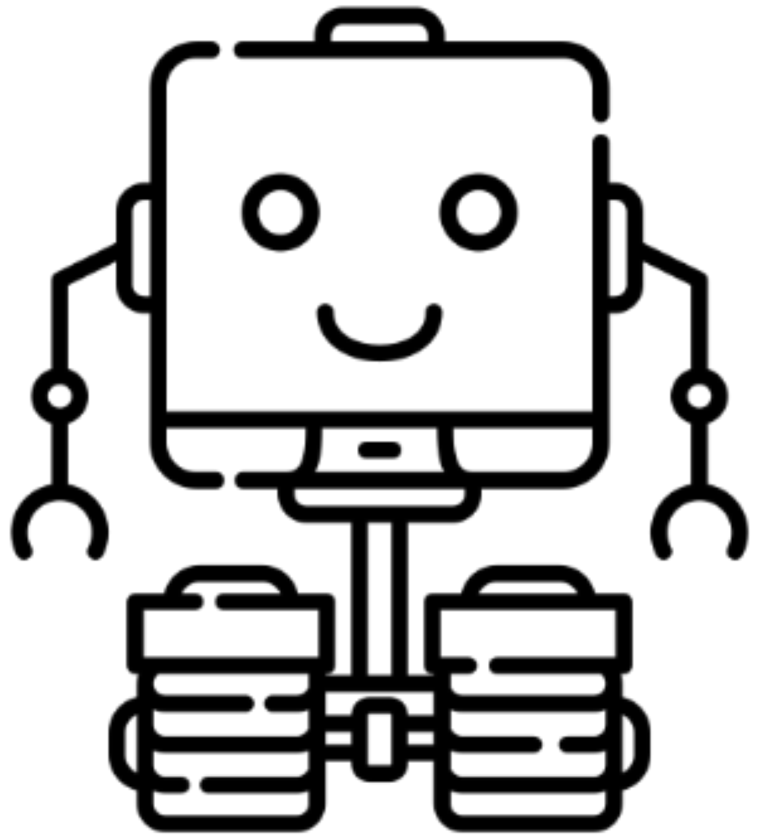
explore

collect log!

[Nottingham et al., Do Embodied Agents Dream of Pixelated Sheep: Embodied Decision Making using Language Guided World Modelling, ICML 2023]

AWM hypothesis testing





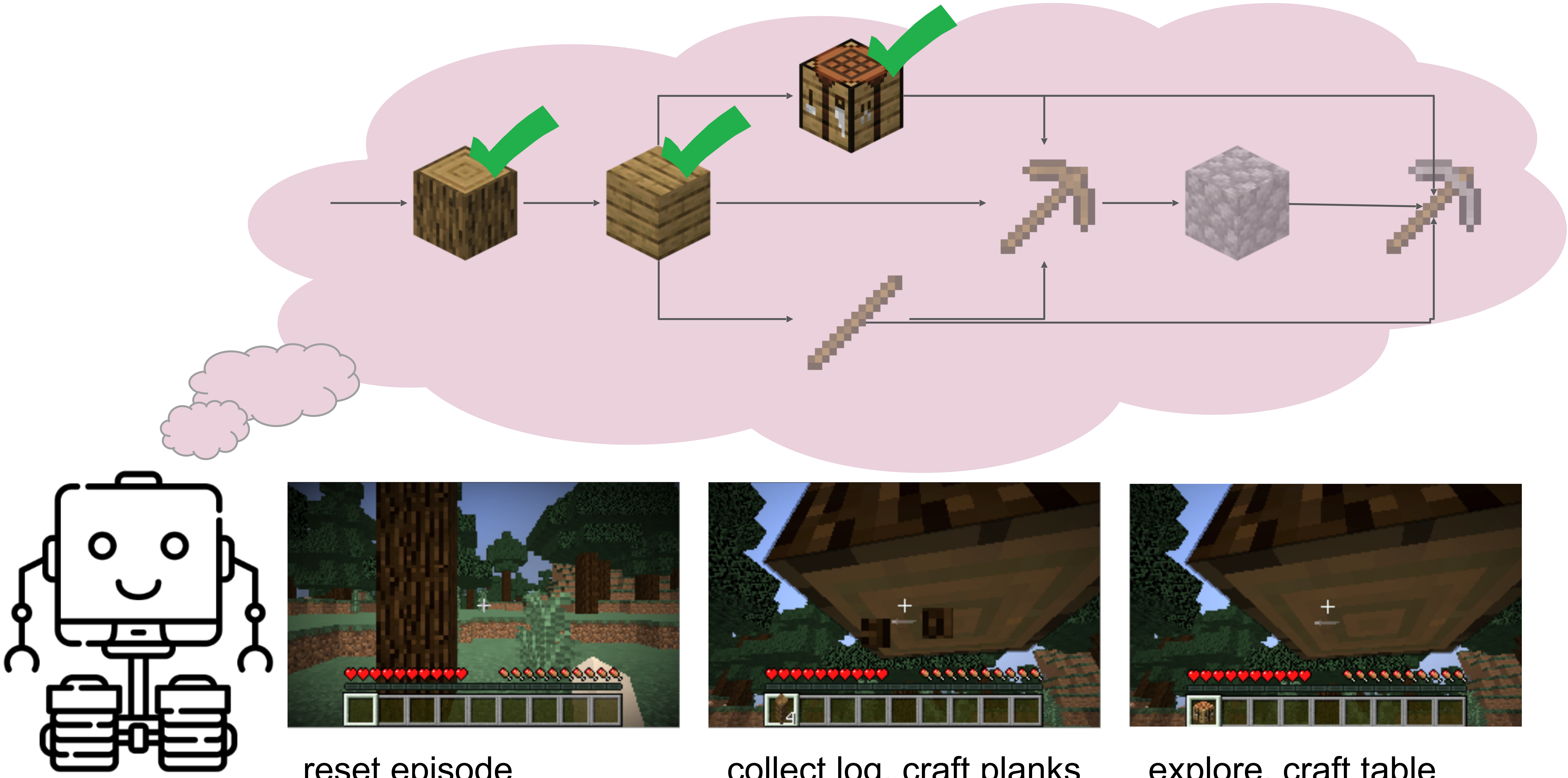
reset episode

collect log

explore, craft planks

[Nottingham et al., Do Embodied Agents Dream of Pixelated Sheep: Embodied Decision Making using Language Guided World Modelling, ICML 2023]

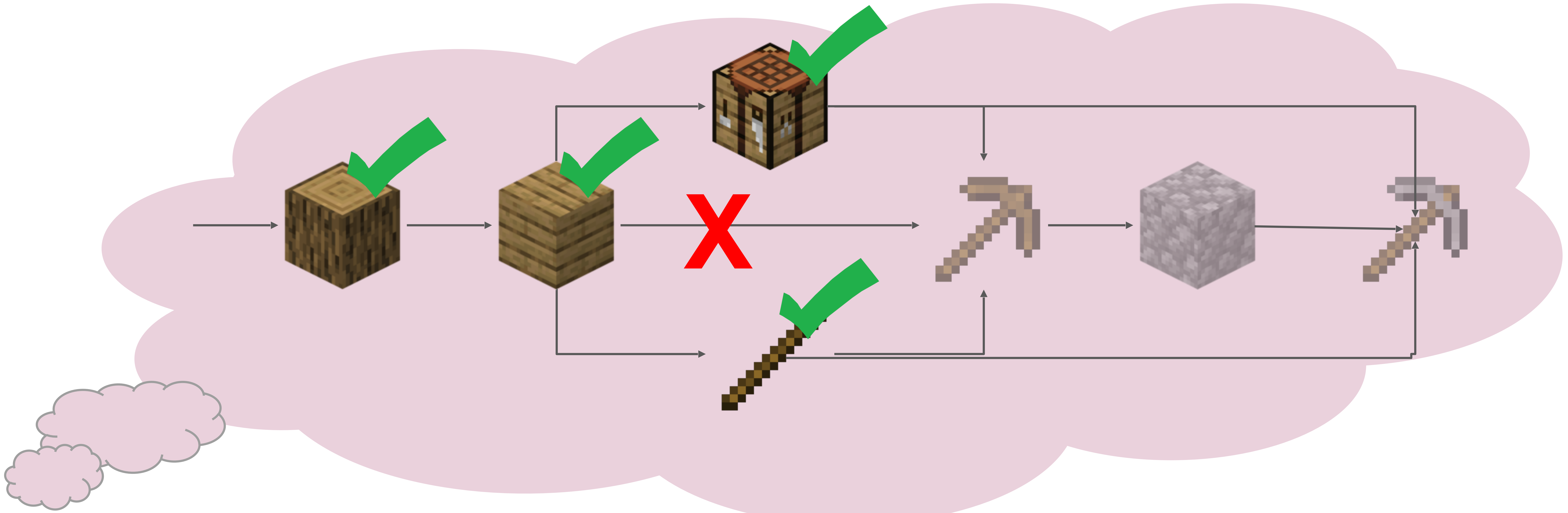
AWM hypothesis testing

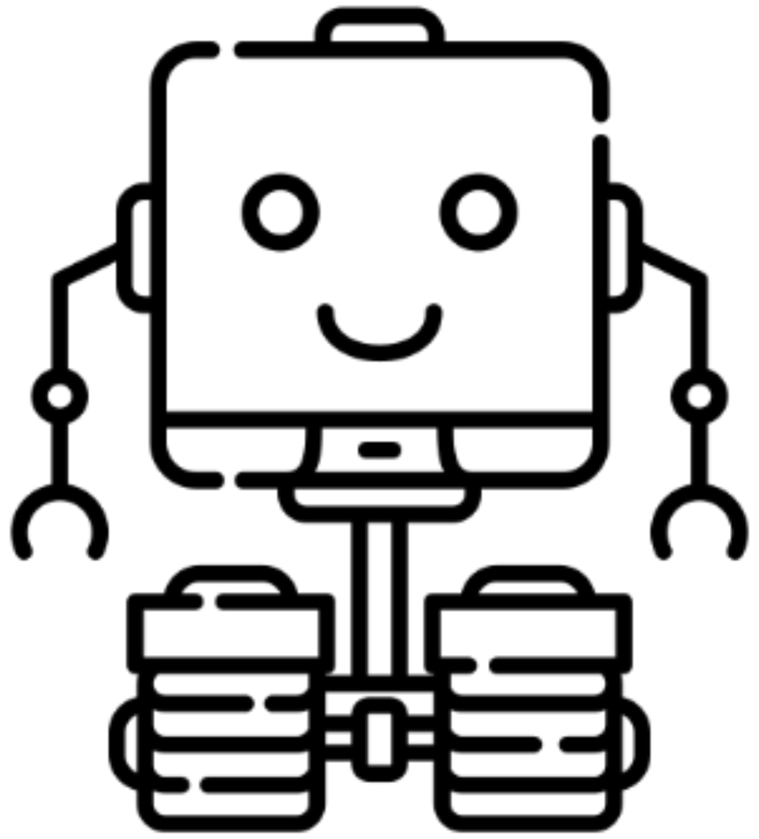


reset episode collect log, craft planks explore, craft table

[Nottingham et al., Do Embodied Agents Dream of Pixelated Sheep: Embodied Decision Making using Language Guided World Modelling, ICML 2023]

AWM hypothesis testing

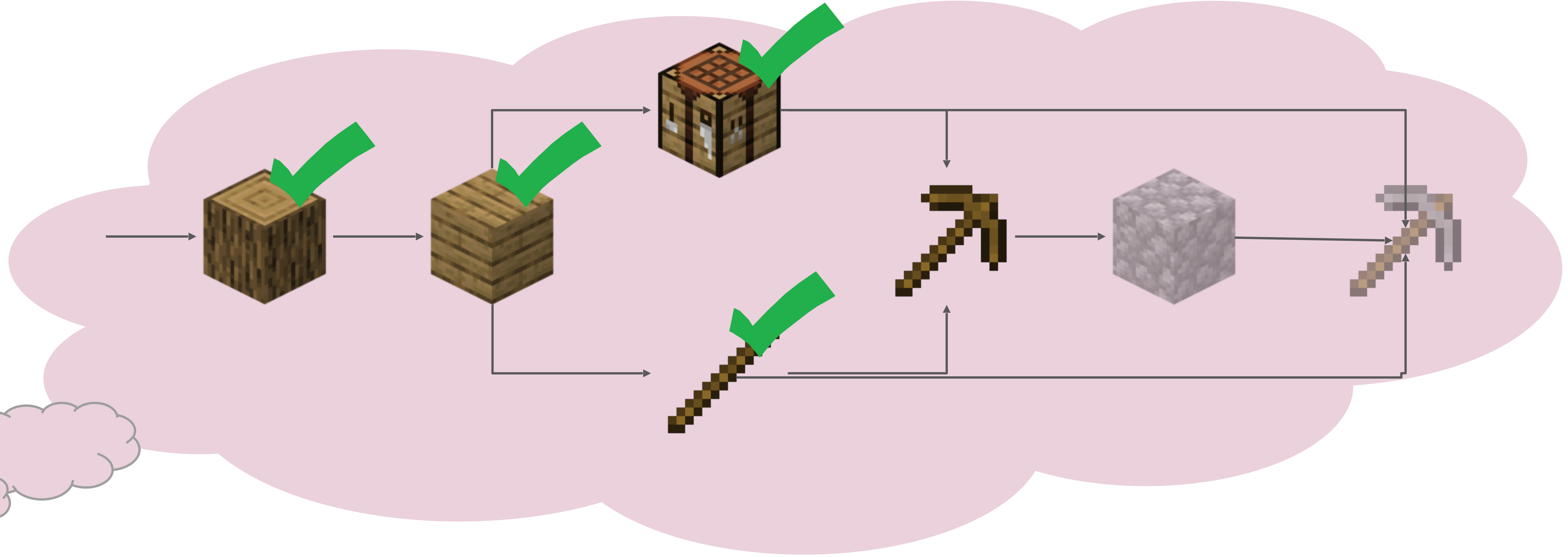


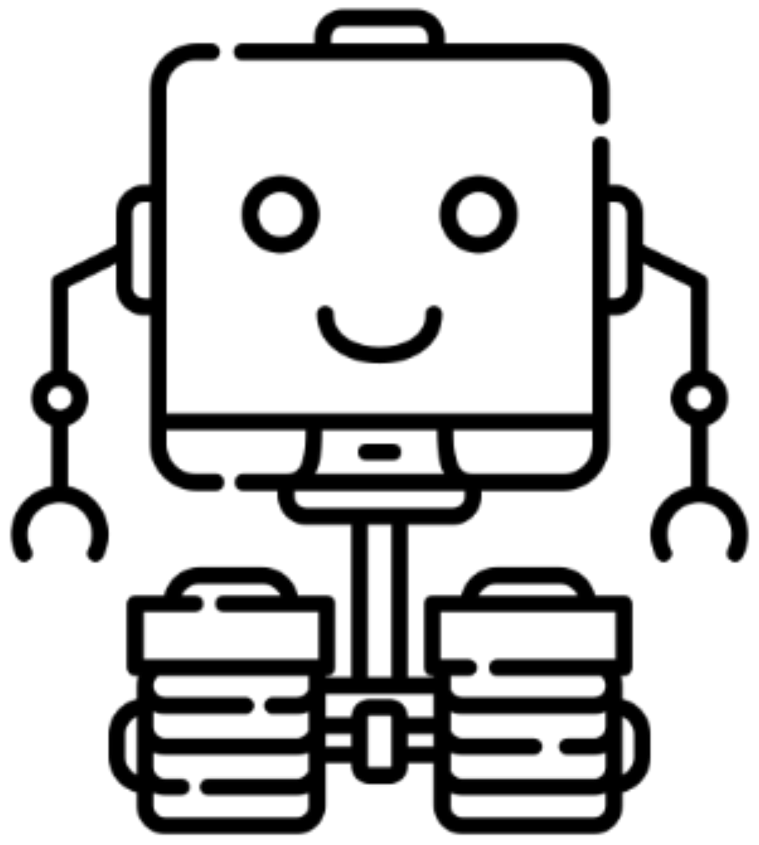


reset episode | collect log, craft planks | explore, craft stick

[Nottingham et al., Do Embodied Agents Dream of Pixelated Sheep: Embodied Decision Making using Language Guided World Modelling, ICML 2023]

AWM hypothesis testing





reset episode

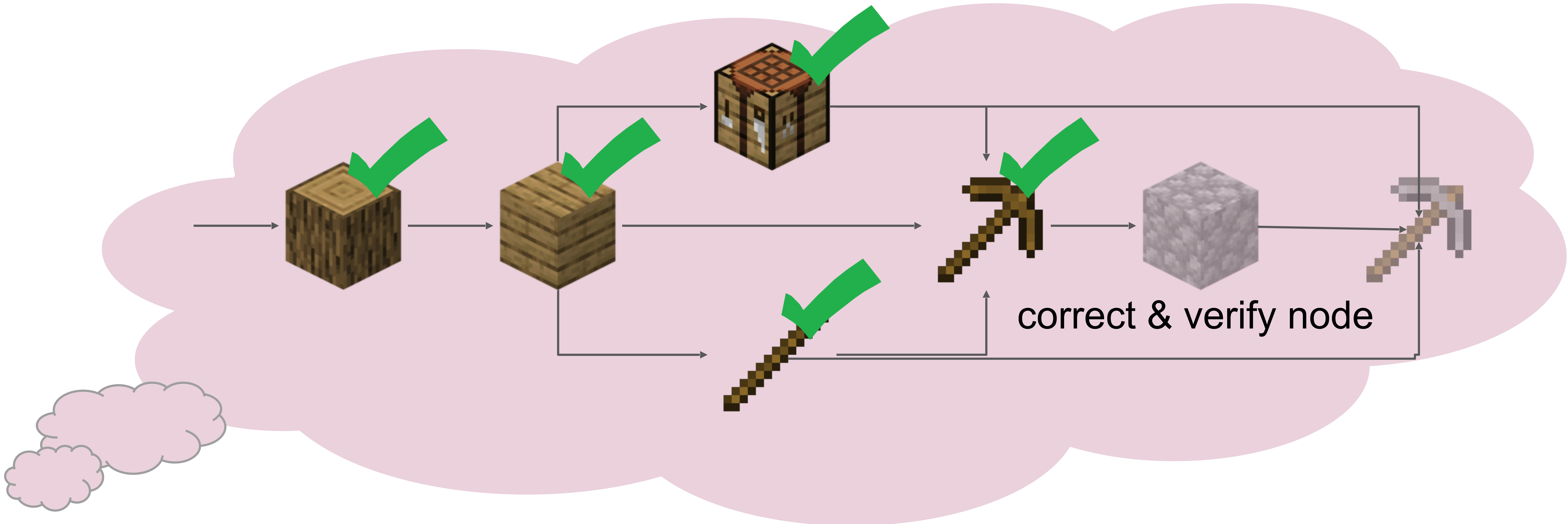
craft dependencies

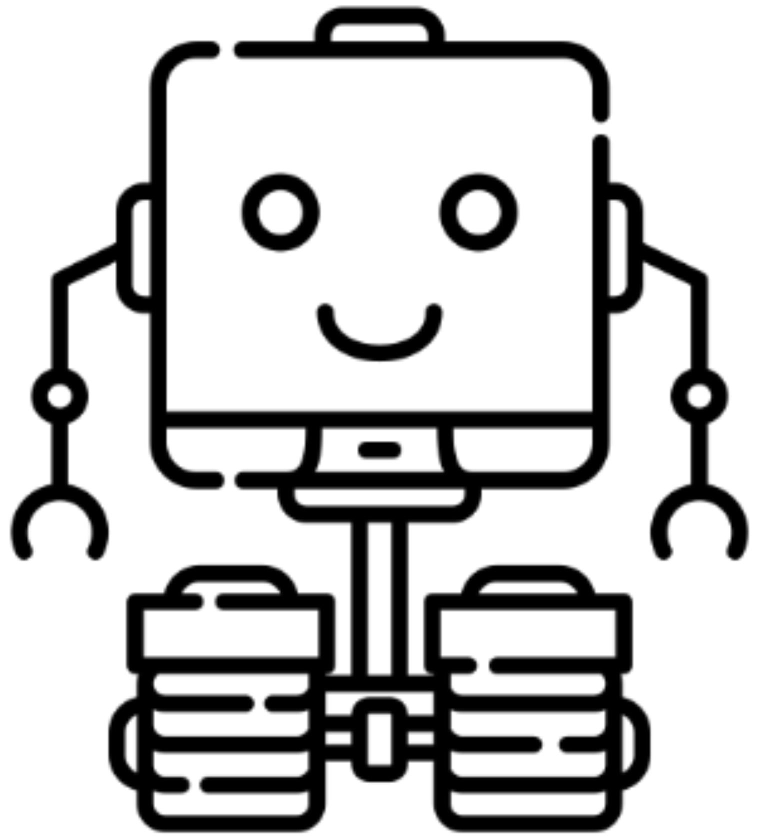
X

failed wooden pickaxe

[Nottingham et al., Do Embodied Agents Dream of Pixelated Sheep: Embodied Decision Making using Language Guided World Modelling, ICML 2023]

AWM hypothesis testing





reset episode

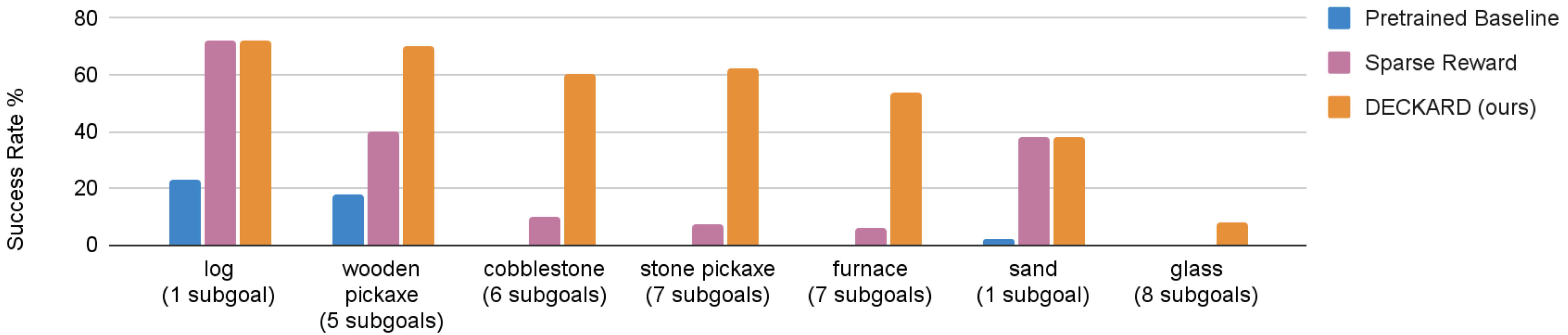
explore

craft wooden pickaxe

[Nottingham et al., Do Embodied Agents Dream of Pixelated Sheep: Embodied Decision Making using Language Guided World Modelling, ICML 2023]

Experiments: task success rate

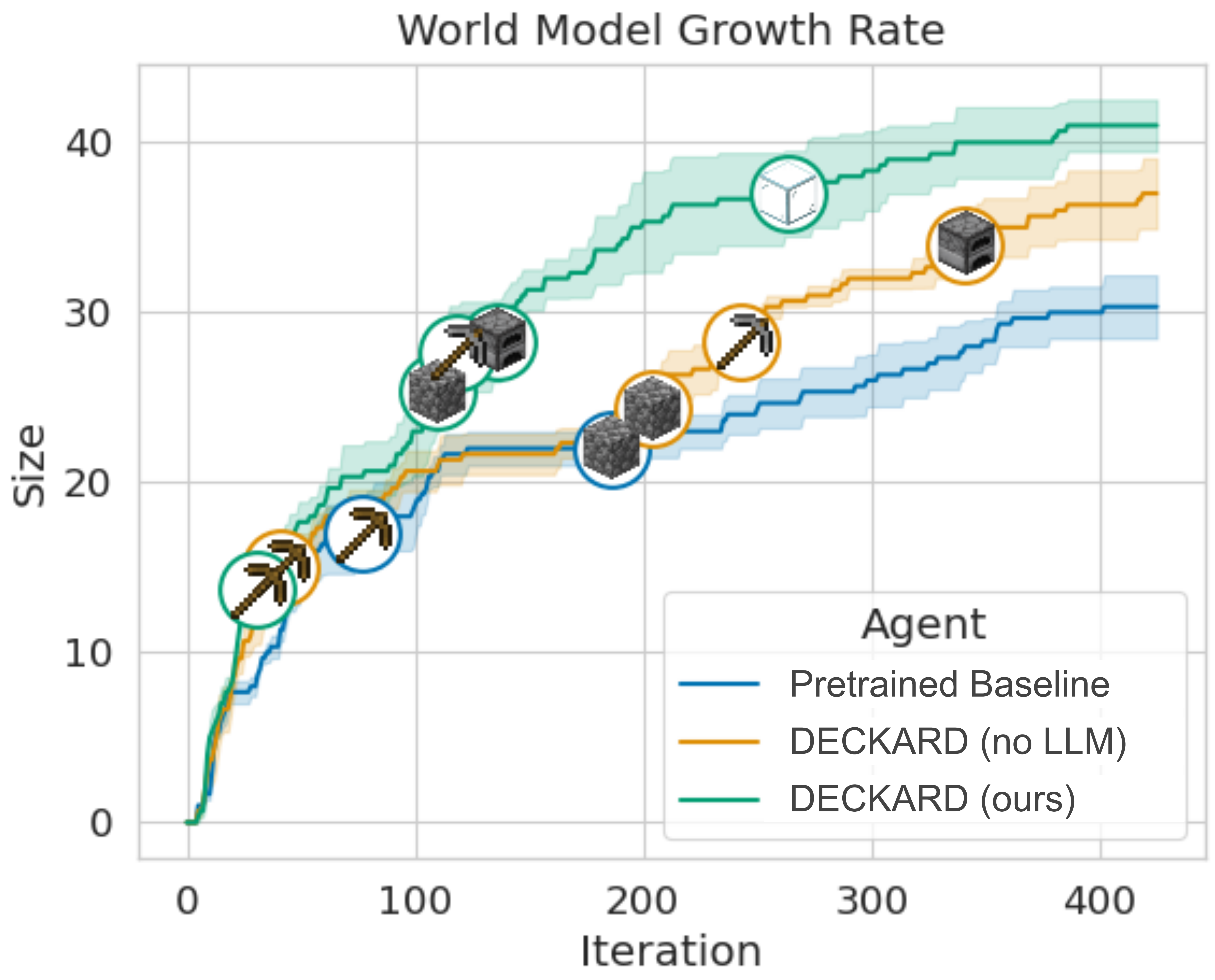
- Success rate of trained policies on specific tasks



[Nottingham et al., Do Embodied Agents Dream of Pixelated Sheep: Embodied Decision Making using Language Guided World Modelling, ICML 2023]

Experiments: exploration rate

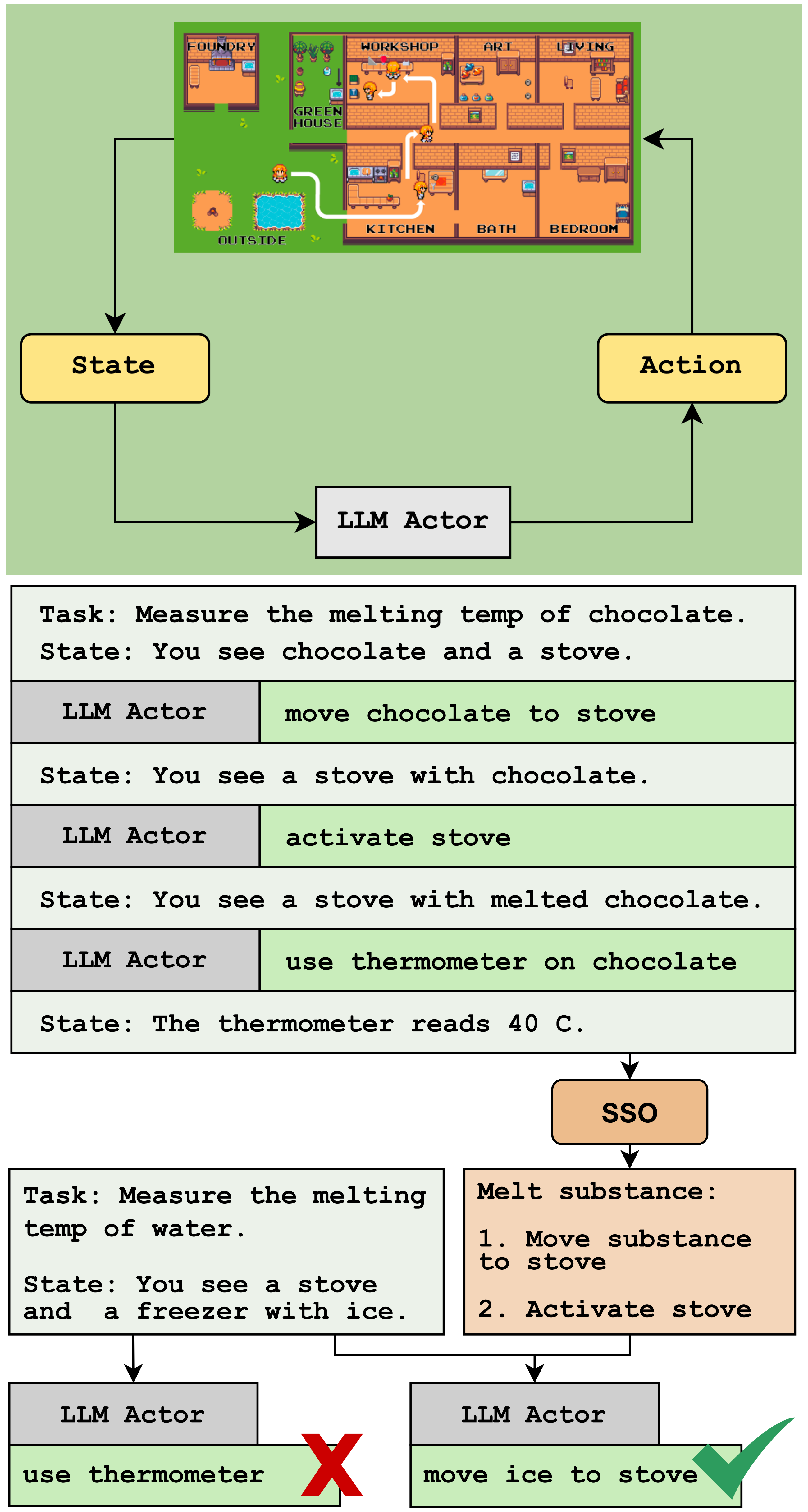
- Discovery of new items when exploring without a task



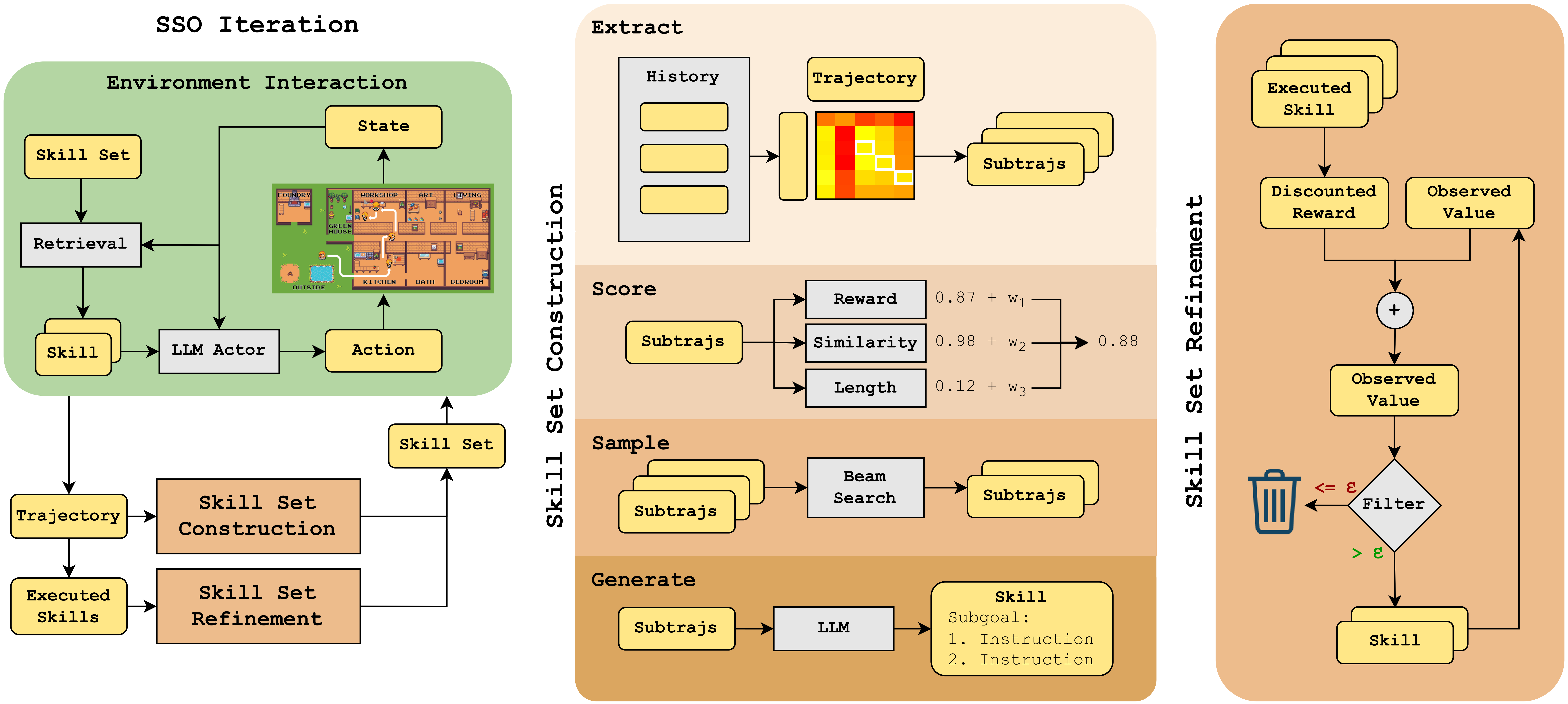
[Nottingham et al., Do Embodied Agents Dream of Pixelated Sheep: Embodied Decision Making using Language Guided World Modelling, ICML 2023]

Enriching the context

- What can we add to the context to help the actor?
  - Using on-task data
- “Is what I'm doing like things I've done before?”
  - Identify behavior prototypes = skills
- Add structure to the actor's execution
  - Higher-level abstractions — semantic and temporal



Skill Set Optimization (SSO)

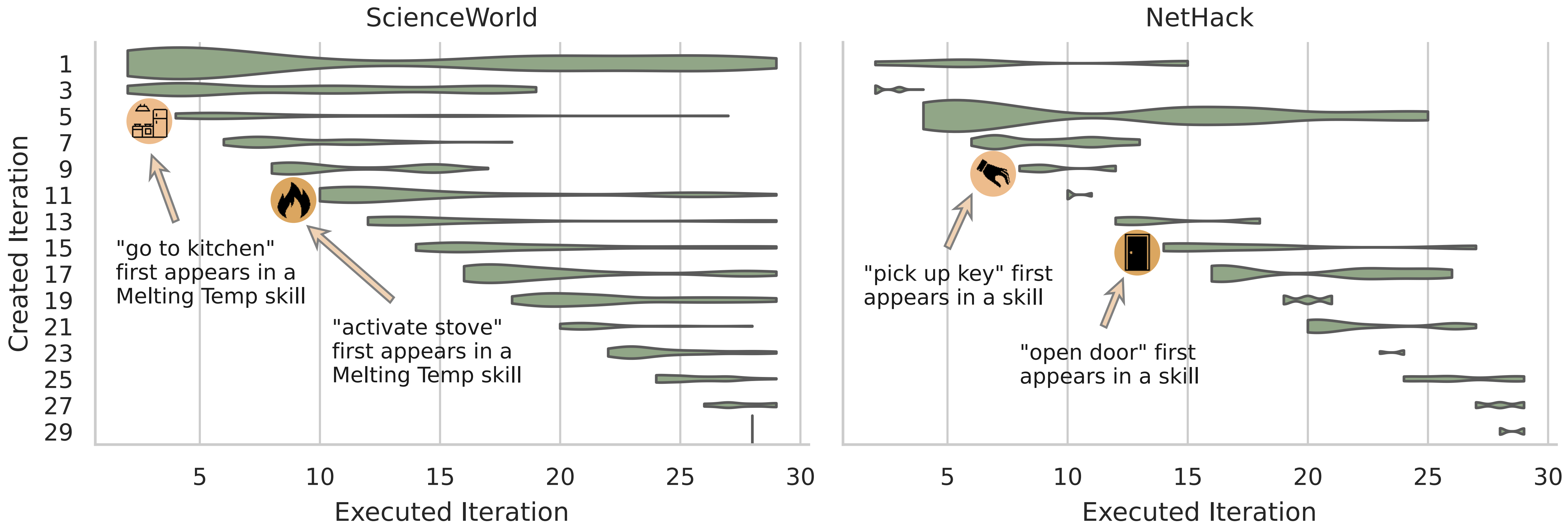


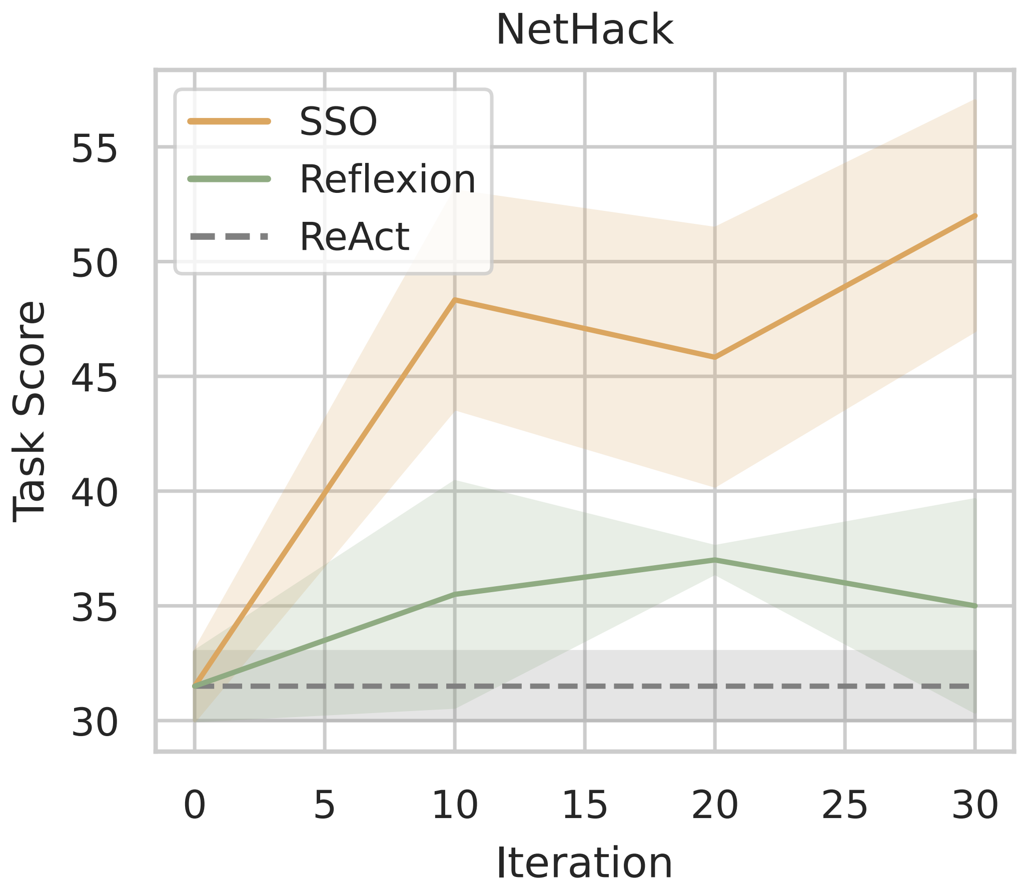
[Nottingham et al., Skill Set Optimization: Reinforcing Language Model Behavior via Transferable Skills, ICML 2024]

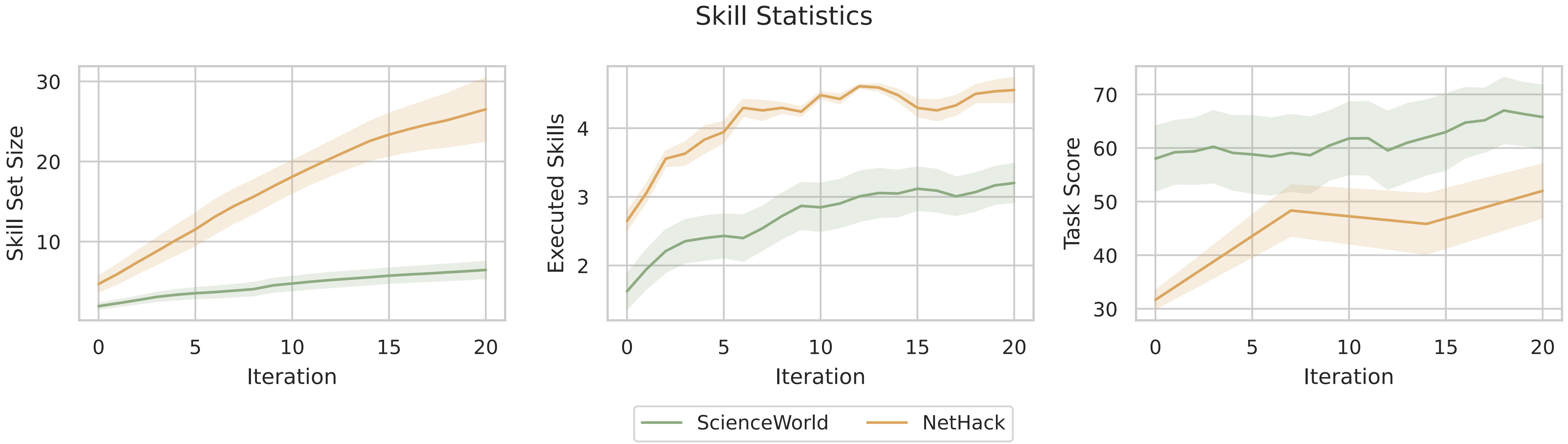
Experiments

ScienceWorld Melting Temp Task
Subgoal: The stove is turned on. on the stove is: a substance called liquid [substance].
1. Focus on the thermometer
2. Focus on the substance you want to heat
3. Move the focused substance to the stove
4. Activate the stove

NetHack Task
Subgoal: You succeed in unlocking the door.
1. Stand adjacent to the closed door that needs to be unlocked
2. Use the action ’a’ to apply the relevant key or tool that can unlock the door
3. Confirm the unlock action by responding affirmatively when prompted, typically by using the action ’y’







[Nottingham et al., Skill Set Optimization: Reinforcing Language Model Behavior via Transferable Skills, ICML 2024]

Recap

- Abstractions: succinct representations; better data efficiency, generalization
- Hierarchical policy is foremost a memory structure
- Structure can be programmed, demonstrated, prompted, or discovered
- Subgoals can be represented by terminal-state value functions
- Many more hierarchical frameworks
- Many more opportunities for structure in control
  - Multi-task learning
  - Structured exploration