# CS 690: Human-Centric Machine Learning

Prof. Scott Niekum

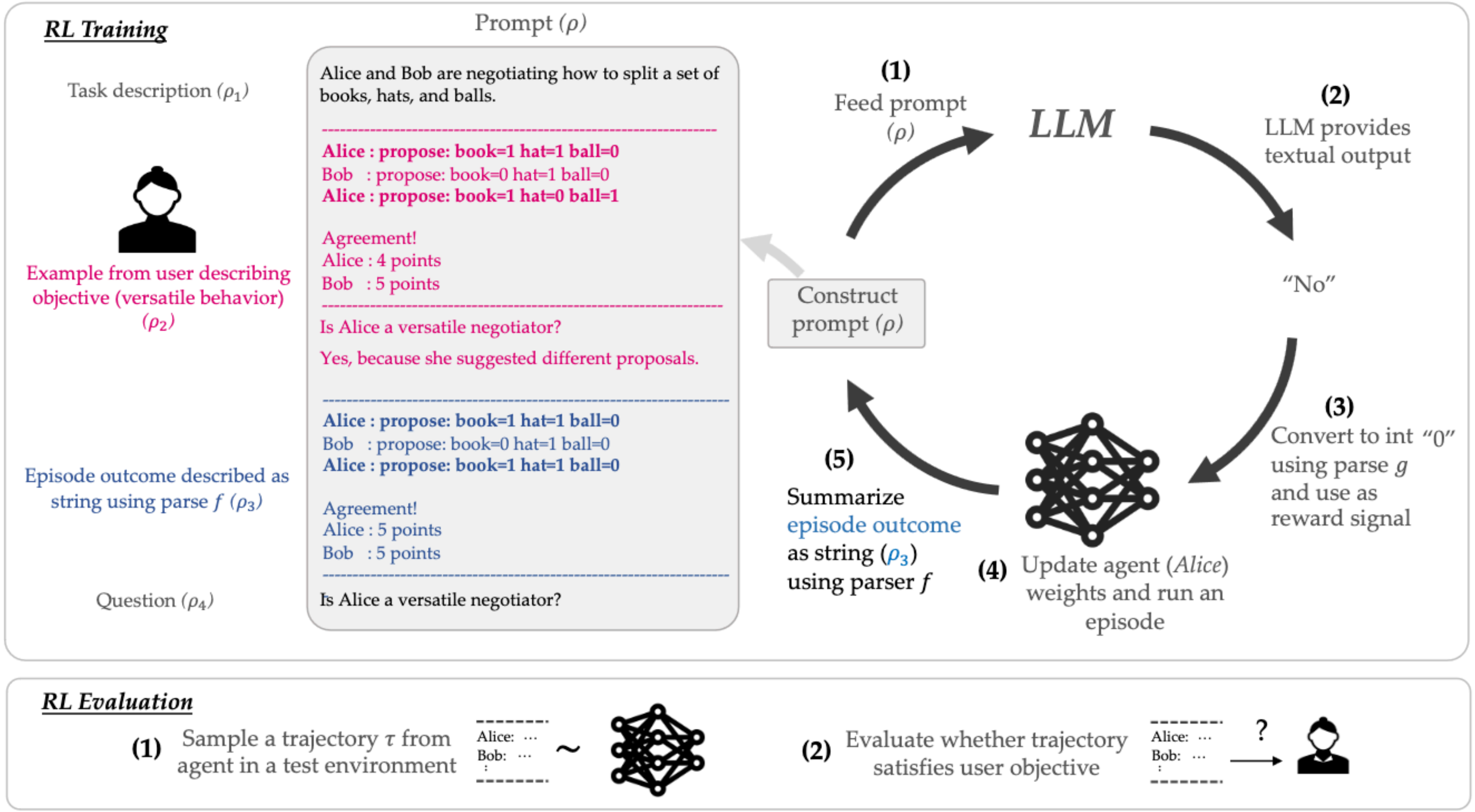
## LLM-based reward design

# Motivation: LLM-based reward design

- We know reward design is hard
- Demonstrations and preferences require models of humans that don't necessarily capture them accurately, and require a lot of data to work well
- Easy for reward inference to overfit and/or mis-align to reward functions that are highly unlikely from a human perspective
- LLMs contain huge amounts of human knowledge, common sense, and priors about things that people generally want
- Can we leverage that to do better?

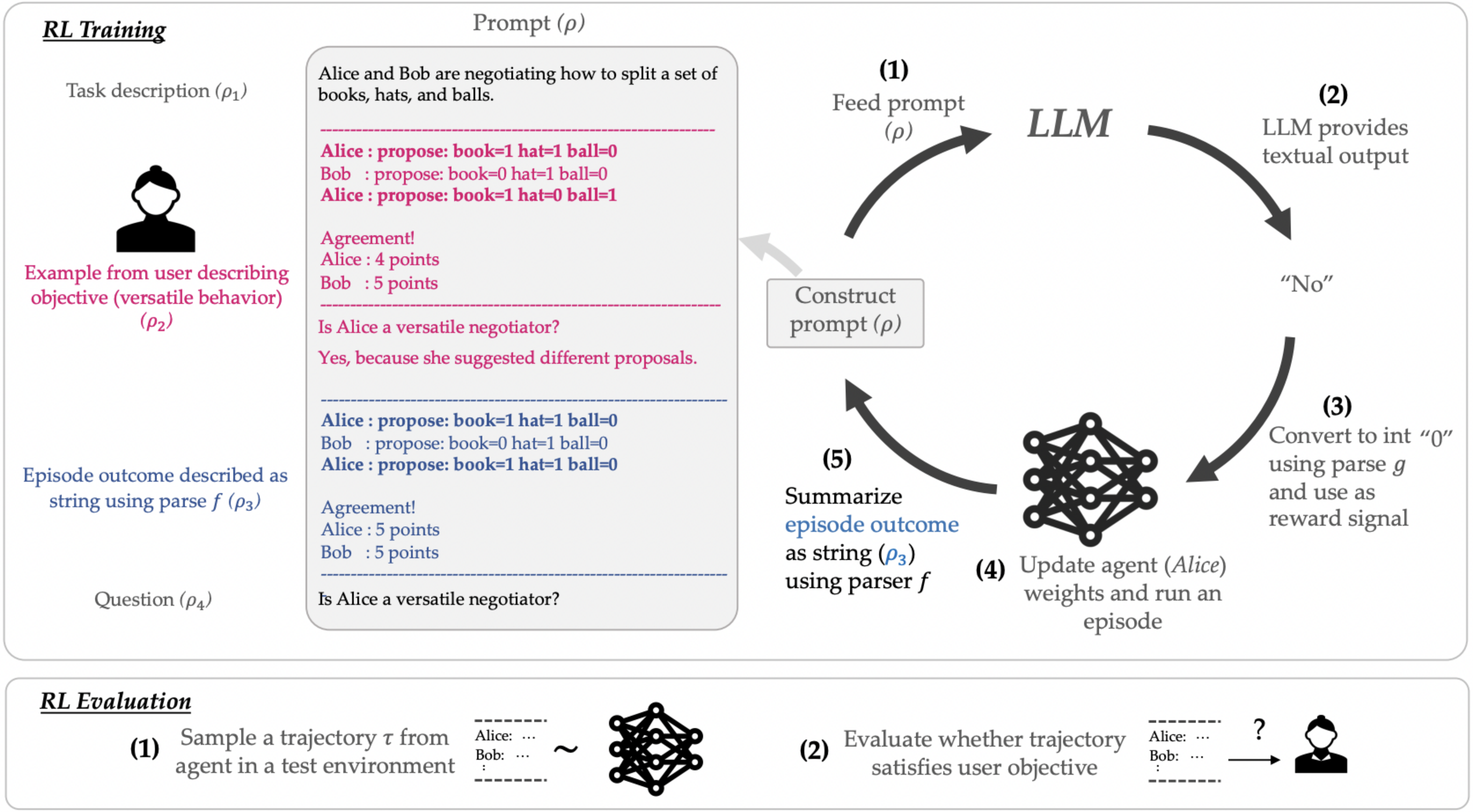
Kwon M, Xie SM, Bullard K, Sadigh D. Reward design with language models. arXiv preprint arXiv:2303.00001. 2023 Feb 27.

# In-context reward learning




Kwon M, Xie SM, Bullard K, Sadigh D. Reward design with language models. arXiv preprint arXiv:2303.00001. 2023 Feb 27.

Slide credit: Sylee Dandekar




Slide credit: Sylee Dandekar

# Ultimatum game

- Proposer (fixed) suggests how to split money and Responder (RL agent) and accept or reject. If reject, no one gets anything
- Rational responder will accept any offer, but if trying to align with user, humans often try to punish an unfair proposer, even if irrational
- They experiment with synthetic users:

  - **Low vs High Percentages.** Users will reject proposals if they receive less than $\{30\%, 60\%\}$ of the endowment.
  - **Low vs High Payoffs.** Users will reject unfair proposals if they receive less than $\{\$10, \$100\}$. They accept unfair proposals otherwise.
  - **Inequity Aversion (Fehr & Schmidt (2010)).** Users will reject proposals if they do not receive exactly 50% of the endowment.

# Ultimatum game

## 10 Examples, No Explanation

*Task description*

P1 and P2 are playing the Ultimatum Game. P1 proposes how they should split $10 and P2 can either accept or reject. If P2 accepts, then the deal is done. If P2 rejects, then both parties get nothing.

*Examples of Objective*

P1 proposes a split of $4.21 for P1 and $5.79 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable?
No

P1 proposes a split of $1.28 for P1 and $8.72 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable?
No

P1 proposes a split of $9.78 for P1 and $0.22 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable?
Yes

[7 more examples]

*Episode outcome*

P1 proposes a split of $9.21 for P1 and $0.79 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable?

*Question*

## 1 Example, with Explanation

*Task description*

P1 and P2 are playing the Ultimatum Game. P1 proposes how they should split $10 and P2 can either accept or reject. If P2 accepts, then the deal is done. If P2 rejects, then both parties get nothing.

*Example of Objective*

P1 proposes a split of $9.78 for P1 and $0.22 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable? Let's think step by step: P2 receives $0.22 < $3 so P2 should reject this offer. Therefore, the outcome is desirable.

*Episode outcome*

P1 proposes a split of $9.21 for P1 and $0.79 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable? Let's think step by step:

*Question*

# Ultimatum game


Figure 2: **Ultimatum Game, Few-shot**. (Top) Accuracy of reward signals provided by LLM and SL during RL training when prompted with/trained on 10 vs 1 example. (Bottom) Corresponding accuracy of RL agents after training. LLM is able to maintain a high accuracy when prompted with a single example followed by an explanation. We do not provide figures of *Inequity Aversion* because both LLM and SL trivially achieve perfect labeling and RL agent accuracy.

# Matrix Game

| | Action 1 | Action 2 |
|---|---|---|
| Action 1 | 2, 1 | 0, 0 |
| Action 2 | 0, 0 | 1, 2 |

**Automated Metrics (Ground Truth Rewards)**

- **Total Welfare**: Outcomes that achieve the greatest sum of player rewards
- **Equality**: Outcomes that result in equal rewards
- **Rawlsian Fairness**: Outcomes that maximize the minimum reward any player can receive
- **Pareto-optimality**: Outcomes where one of the corresponding rewards cannot be improved without lowering the other

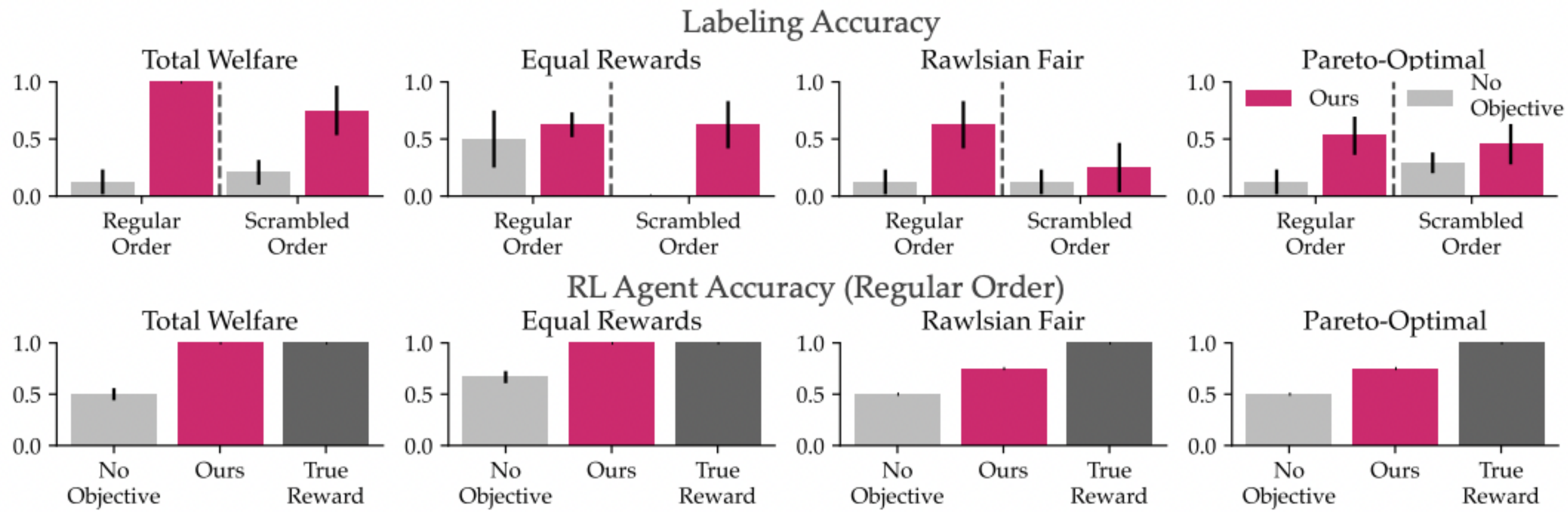
# Matrix Game

| | Total Welfare | Equality | Rawlsian Fairness | Pareto-optimality | No Objective |
|---|---|---|---|---|---|
| *Task description* | We have a two-player game where P1 and P2 can choose one of these options.<br>Options:<br>A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2.<br>B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3.<br>C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1.<br>D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0. | We have a two-player game where P1 and P2 can choose one of these options.<br>Options:<br>A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2.<br>B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3.<br>C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1.<br>D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0. | We have a two-player game where P1 and P2 can choose one of these options.<br>Options:<br>A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2.<br>B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3.<br>C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1.<br>D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0. | We have a two-player game where P1 and P2 can choose one of these options.<br>Options:<br>A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2.<br>B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3.<br>C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1.<br>D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0. | We have a two-player game where P1 and P2 can choose one of these options.<br>Options:<br>A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2.<br>B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3.<br>C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1.<br>D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0. |
| *Question* | Which option(s) result in the greatest total welfare? Let's think step by step: *Description of Objective*<br>Total welfare is | Which option(s) result in equality of rewards? Let's think step by step:<br>Equality of rewards is | Which option(s) result in Rawlsian fair rewards? Let's think step by step:<br>Rawlsian fairness is | Which option(s) are Pareto-optimal? Let's think step by step:<br>An outcome is Pareto-optimal if | Which option(s) should P1 and P2 select? |

# Matrix Game

Labeling accuracy:
- Regular order: Outperforms by 48% on average
- Scrambled order: Outperforms by 36% on average

| | Action 1 | Action 2 |
|---|---|---|
| Action 1 | 2, 1 | 0, 0 |
| Action 2 | 0, 0 | 1, 2 |




Slide credit: Sylee Dandekar

# DealOrNoDeal Task

Baseline: Supervised learning model trained to predict reward, given same examples.

Negotiation Styles:

1. Versatile
2. Push Over
3. Competitive
4. Stubborn

---

**Alice : propose: book=1 hat=1 ball=0**
Bob : propose: book=0 hat=1 ball=0
**Alice : propose: book=1 hat=0 ball=1**

Agreement!
Alice : 4 points
Bob : 5 points

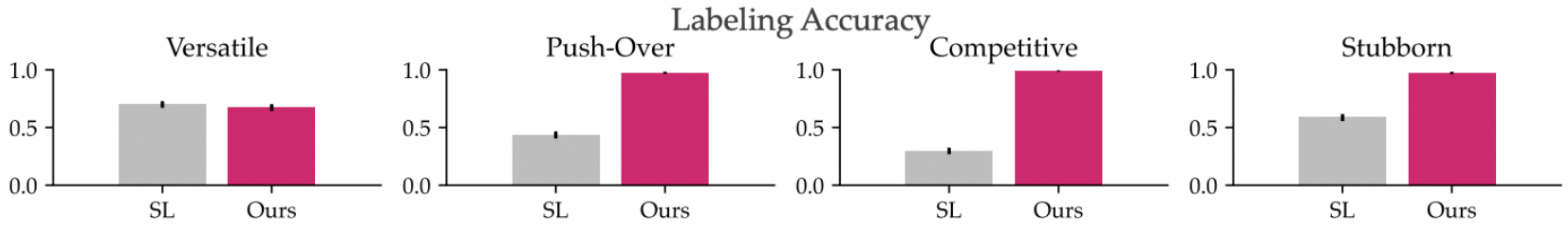
---

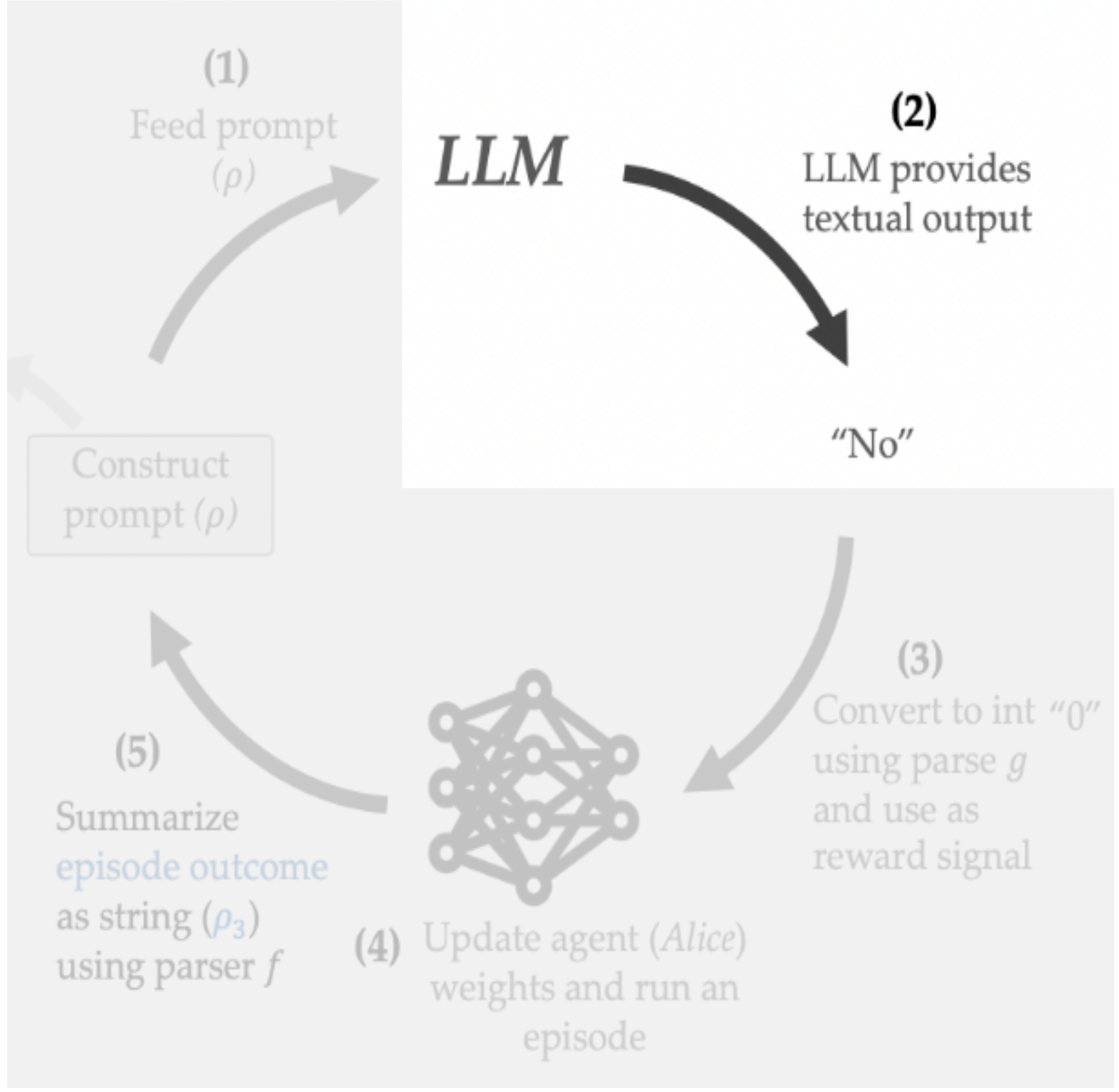
Is Alice a versatile negotiator?

Yes, because she suggested different proposals.

Slide credit: Sylee Dandekar

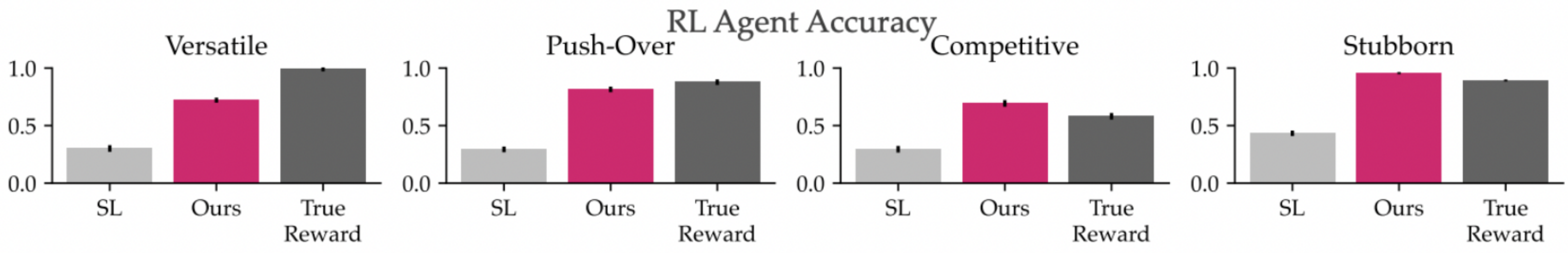
# DealOrNoDeal Task: LLM Accuracy




Slide credit: Sylee Dandekar

# DealOrNoDeal Task: RL Accuracy




- Outperforms SL by an average of 46%
- Underperforms ground truth by an average of 4%

Slide credit: Sylee Dandekar

# DealOrNoDeal Task: Human Evaluation

- No ground truth rewards
- Human evaluators

```
=================================================================================
Alice : book=(count:1 value:4) hat=(count:1 value:3) ball=(count:3 value:1)
Bob   : book=(count:1 value:6) hat=(count:1 value:4) ball=(count:3 value:0)
---------------------------------------------------------------------------------
Bob   : propose: book=1 hat=1 ball=0
Alice : propose: book=1 hat=1 ball=3
Bob   : propose: book=1 hat=1 ball=0
Alice : propose: book=1 hat=1 ball=3
Bob   : propose: book=1 hat=1 ball=0
Alice : propose: book=1 hat=1 ball=3
Bob   : disagree
Alice : insist: book=0 hat=1 ball=3
Bob   : agree
---------------------------------------------------------------------------------
Agreement!
Alice : 6 points
Bob   : 6 points

Is Alice an ambitious negotiator?

Yes, because Alice did not let the deal end into disagreement and insisted on finding a solution.
```

```
=================================================================================
Alice : book=(count:1 value:4) hat=(count:3 value:1) ball=(count:1 value:3)
Bob   : book=(count:1 value:9) hat=(count:3 value:0) ball=(count:1 value:1)
---------------------------------------------------------------------------------
Bob   : propose: book=1 hat=0 ball=1
Alice : propose: book=1 hat=3 ball=0
Bob   : propose: book=1 hat=0 ball=0
Alice : propose: book=1 hat=3 ball=1
Bob   : propose: book=1 hat=0 ball=0
Alice : propose: book=1 hat=3 ball=1
Bob   : propose: book=1 hat=0 ball=0
Alice : propose: book=1 hat=0 ball=0
Bob   : propose: book=1 hat=0 ball=0
Alice : propose: book=1 hat=0 ball=0
Bob   : propose: book=1 hat=0 ball=0
Alice : propose: book=1 hat=0 ball=0
Bob   : propose: book=1 hat=0 ball=0
---------------------------------------------------------------------------------
Disagreement?!
Alice : 0 points
Bob   : 0 points

Is Alice an ambitious negotiator?
Yes, because Alice wanted her most valued item, and took the risk of getting into a disagreement.
```

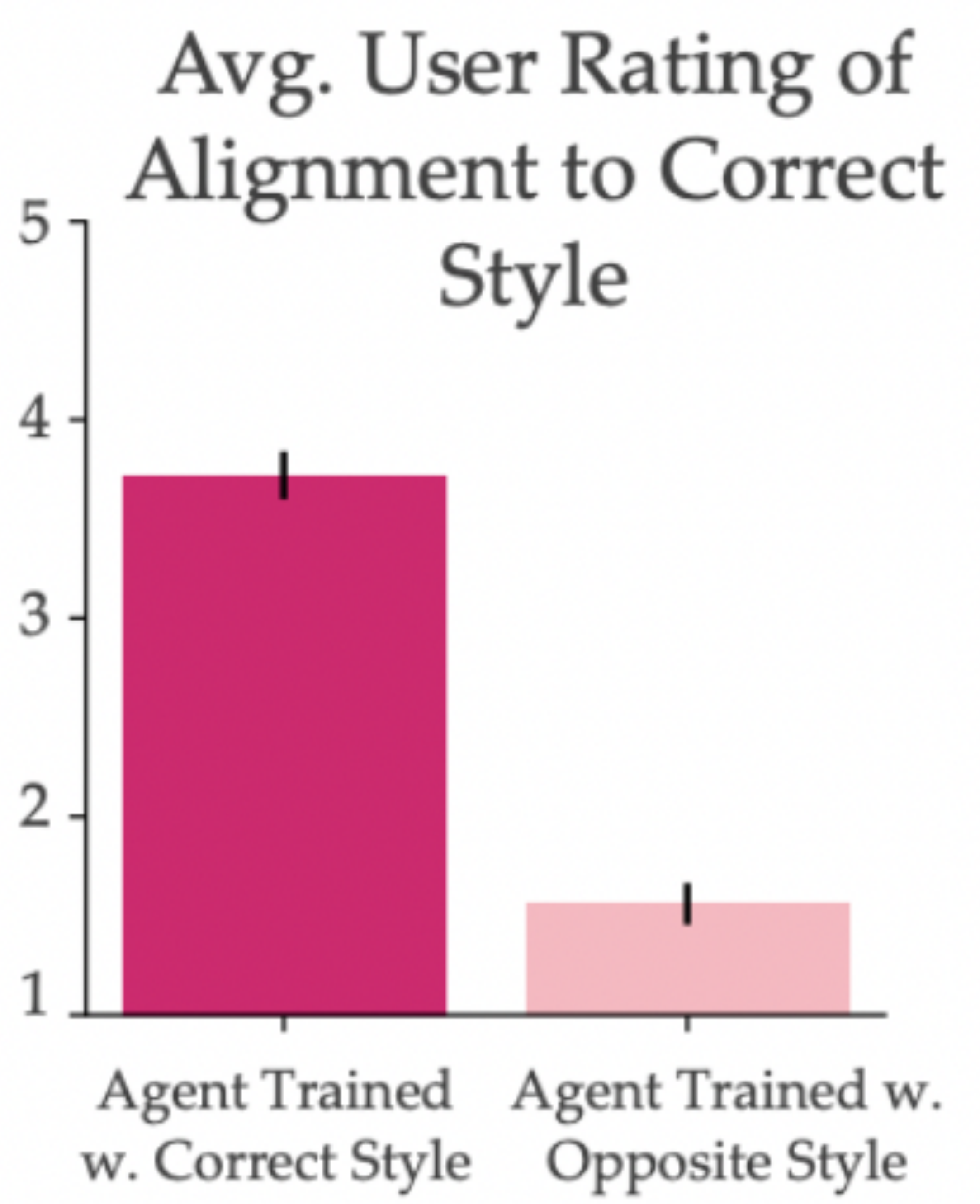
```
=================================================================================
Alice : book=(count:3 value:1) hat=(count:1 value:1) ball=(count:1 value:6)
Bob   : book=(count:3 value:0) hat=(count:1 value:5) ball=(count:1 value:5)
---------------------------------------------------------------------------------
Alice : propose: book=3 hat=0 ball=1
Bob   : propose: book=0 hat=1 ball=1
Alice : agree
---------------------------------------------------------------------------------
Agreement!
Alice : 3 points
Bob   : 10 points
=================================================================================

Is Alice an ambitious negotiator?

No, because she did not propose a counter-offer to Bob's bad offer.
```

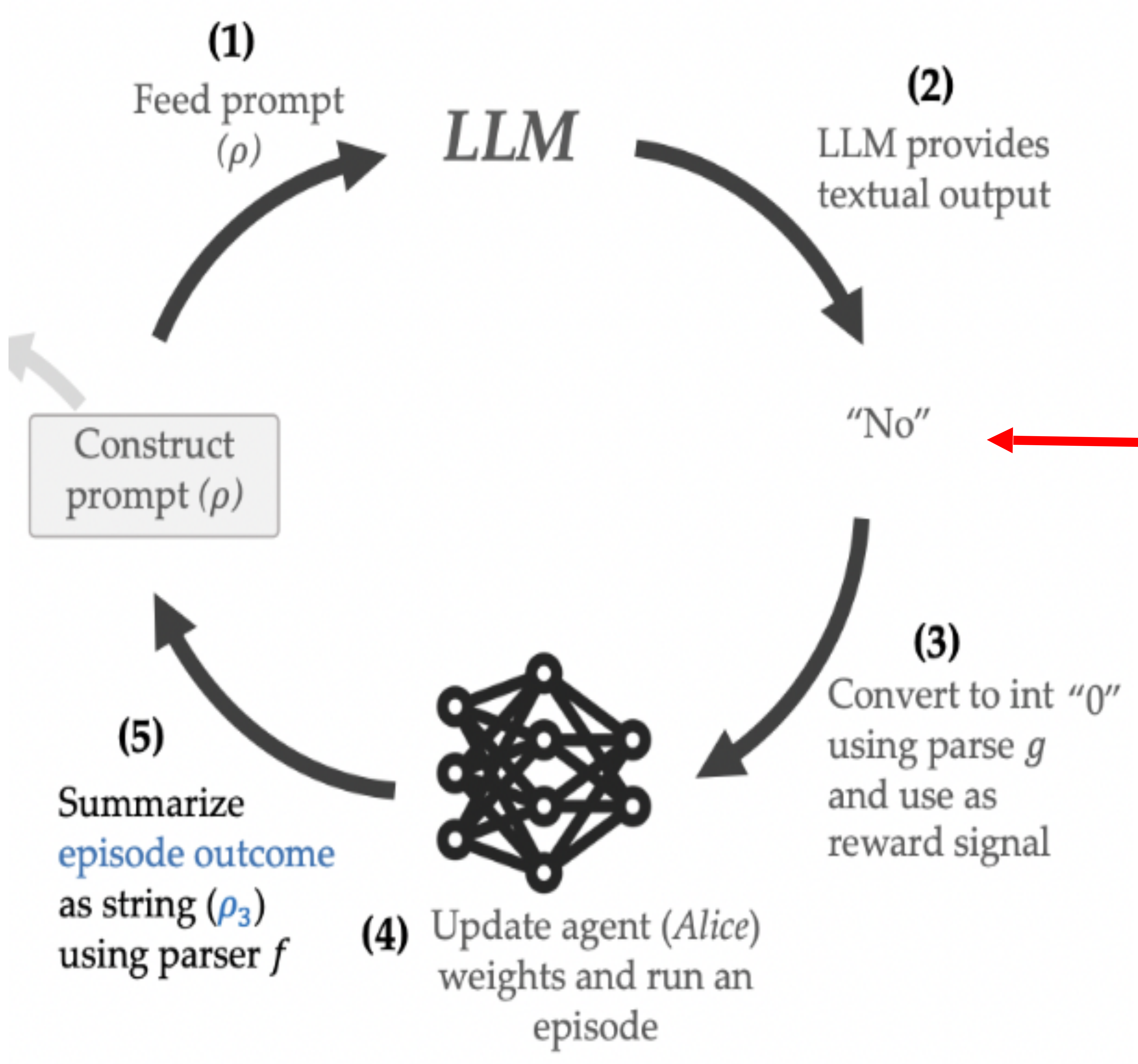
Slide credit: Sylee Dandekar

# DealOrNoDeal Task: Human Evaluation




**(Only 10 participants)**

# Limitations




Requires text based inputs

Only used binary rewards

# Assumptions

- The prior paper assumed existence a parser
- Only worked with binary rewards
- Not interpreatable

# Eureka

Ma YJ, Liang W, Wang G, Huang DA, Bastani O, Jayaraman D, Zhu Y, Fan L, Anandkumar A. Eureka: Human-level reward design via coding large language models. arXiv preprint arXiv:2310.12931. 2023 Oct 19.

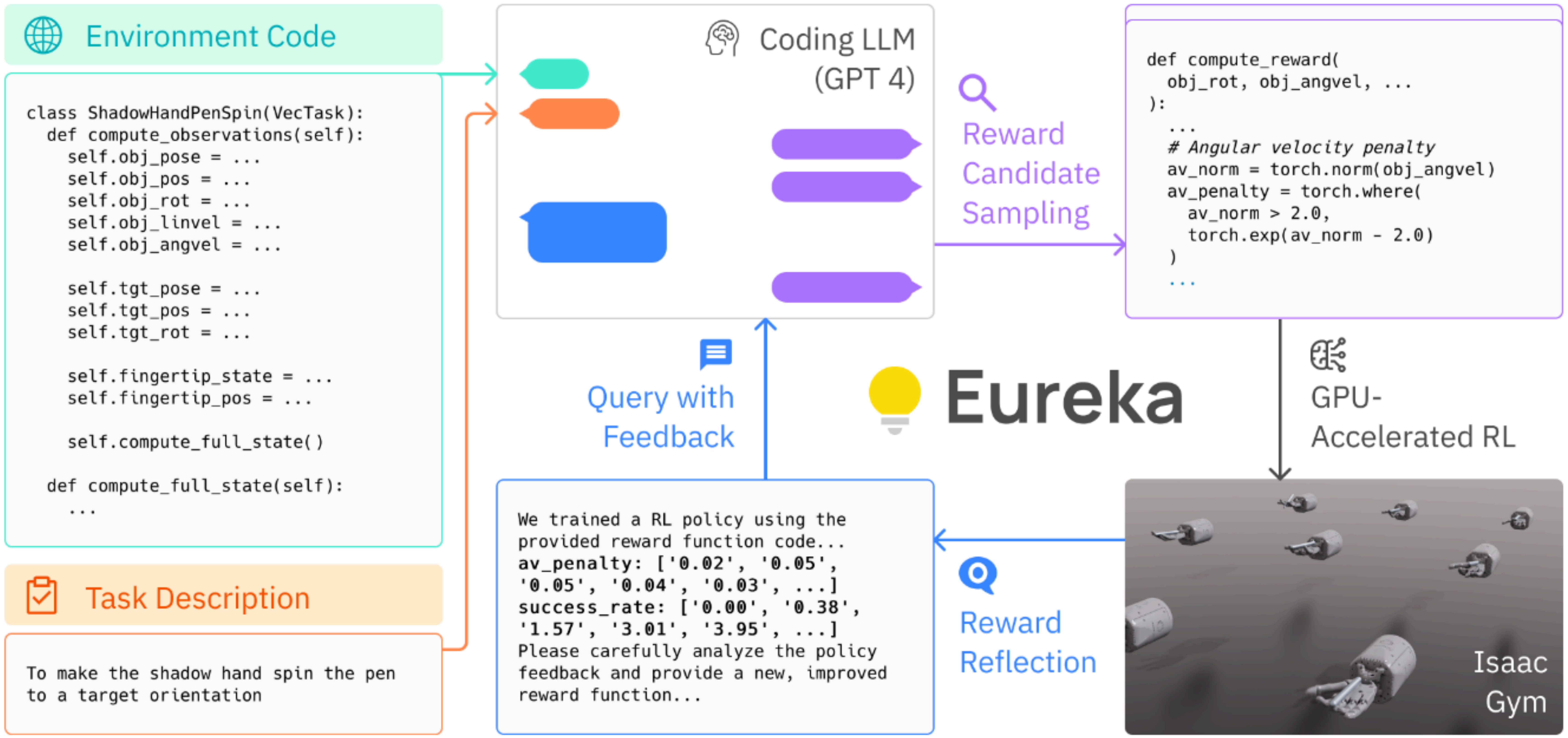
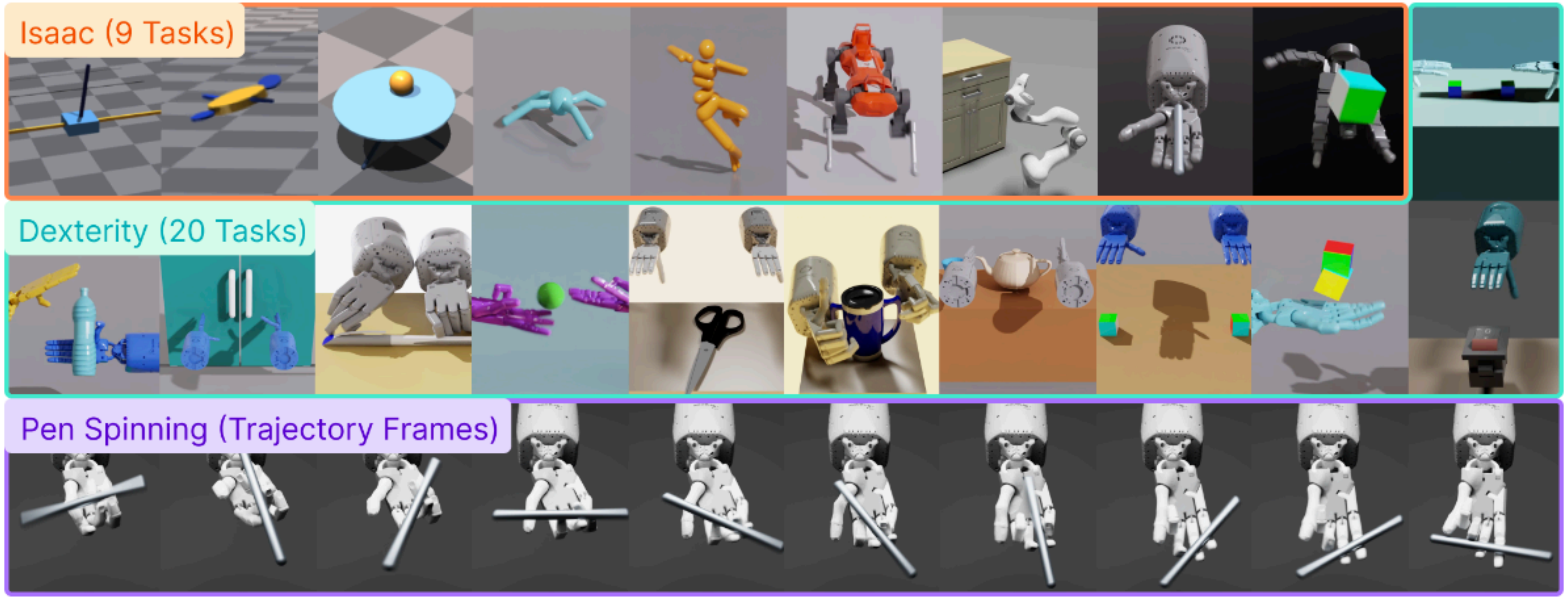



Figure 2: EUREKA takes unmodified environment source code and language task description as context to zero-shot generate executable reward functions from a coding LLM. Then, it iterates between reward sampling, GPU-accelerated reward evaluation, and reward reflection to progressively improve its reward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

# Eureka



Isaac (9 Tasks)
Dexterity (20 Tasks)
Pen Spinning (Trajectory Frames)

# Eureka

**Algorithm 1** EUREKA

**Require**: Task description $l$, environment code $M$,
coding LLM `LLM`, fitness function $F$, initial prompt `prompt`
**Hyperparameters**: search iteration $N$, iteration batch size $K$
**for** N iterations **do**
    // Sample $K$ reward code from LLM
    $\mathtt{R}_1, ..., \mathtt{R}_k \sim \mathtt{LLM}(l, M, \mathtt{prompt})$
    // Evaluate reward candidates
    $s_1 = F(\mathtt{R}_1), ..., s_K = F(\mathtt{R}_K)$
    // Reward reflection
    $\mathtt{prompt} := \mathtt{prompt} : \mathtt{Reflection}(R^n_{best}, s^n_{best})$,
    where $best = \arg\max_k s_1, ..., s_K$
    // Update Eureka reward
    $R_{\text{Eureka}}, s_{\text{Eureka}} = (R^n_{\text{best}}, s^n_{\text{best}}), \quad \text{if } s^n_{\text{best}} > s_{\text{Eureka}}$
**Output**: $R_{\text{Eureka}}$

# Eureka

Prompt 1: Initial system prompt

```
You are a reward engineer trying to write reward functions to solve reinforcement learning
     tasks as effective as possible.
Your goal is to write a reward function for the environment that will help the agent learn the
      task described in text.
Your reward function should use useful variables from the environment as inputs. As an example
     ,
the reward function signature can be:
@torch.jit.script
def compute_reward(object_pos: torch.Tensor, goal_pos: torch.Tensor) -> Tuple[torch.Tensor,
     Dict[str, torch.Tensor]]:
    ...
    return reward, {}
Since the reward function will be decorated with @torch.jit.script,
please make sure that the code is compatible with TorchScript (e.g., use torch tensor instead
     of numpy array).
Make sure any new tensor or variable you introduce is on the same device as the input tensors.
```

# Eureka

Prompt 3: Code formatting tip

```
The output of the reward function should consist of two items:
    (1) the total reward,
    (2) a dictionary of each individual reward component.
The code output should be formatted as a python code string: "```python ... ```".

Some helpful tips for writing the reward function code:
    (1) You may find it helpful to normalize the reward to a fixed range by applying
    transformations like torch.exp to the overall reward or its components
    (2) If you choose to transform a reward component, then you must also introduce a
    temperature parameter inside the transformation function; this parameter must be a named
    variable in the reward function and it must not be an input variable. Each transformed
    reward component should have its own temperature variable
    (3) Make sure the type of each input variable is correctly specified; a float input
    variable should not be specified as torch.Tensor
    (4) Most importantly, the reward code's input variables must contain only attributes of
    the provided environment class definition (namely, variables that have prefix self.).
    Under no circumstance can you introduce new input variables.
```

# Eureka

Prompt 2: Reward reflection and feedback

```
We trained a RL policy using the provided reward function code and tracked the values of the
     individual components in the reward function as well as global policy metrics such as
     success rates and episode lengths after every {epoch_freq} epochs and the maximum, mean,
     minimum values encountered:
<REWARD REFLECTION HERE>

Please carefully analyze the policy feedback and provide a new, improved reward function that
     can better solve the task. Some helpful tips for analyzing the policy feedback:
     (1) If the success rates are always near zero, then you must rewrite the entire reward
     function
     (2) If the values for a certain reward component are near identical throughout, then this
     means RL is not able to optimize this component as it is written. You may consider
         (a) Changing its scale or the value of its temperature parameter
         (b) Re-writing the reward component
         (c) Discarding the reward component
     (3) If some reward components' magnitude is significantly larger, then you must re-scale
     its value to a proper range
Please analyze each existing reward component in the suggested manner above first, and then
     write the reward function code.
```

# Eureka

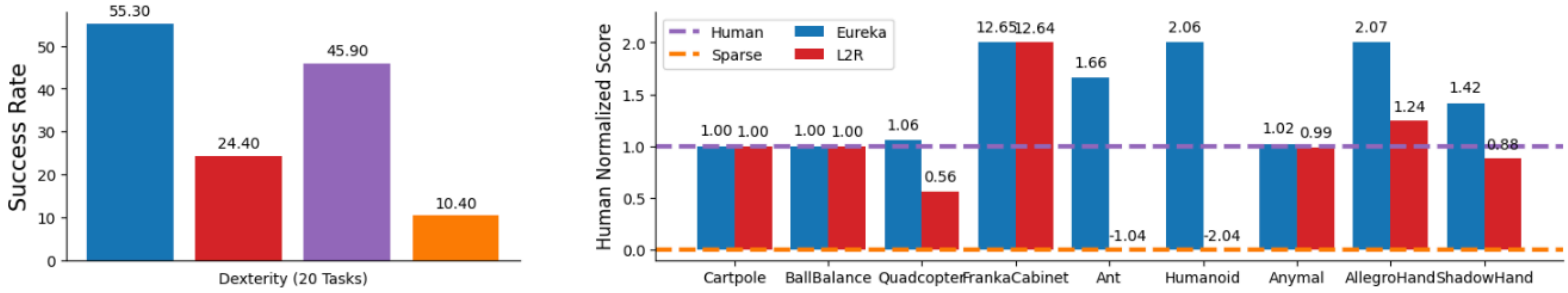



Figure 4: EUREKA outperforms Human and L2R across all tasks. In particular, EUREKA realizes much greater gains on high-dimensional dexterity environments.

# Eureka

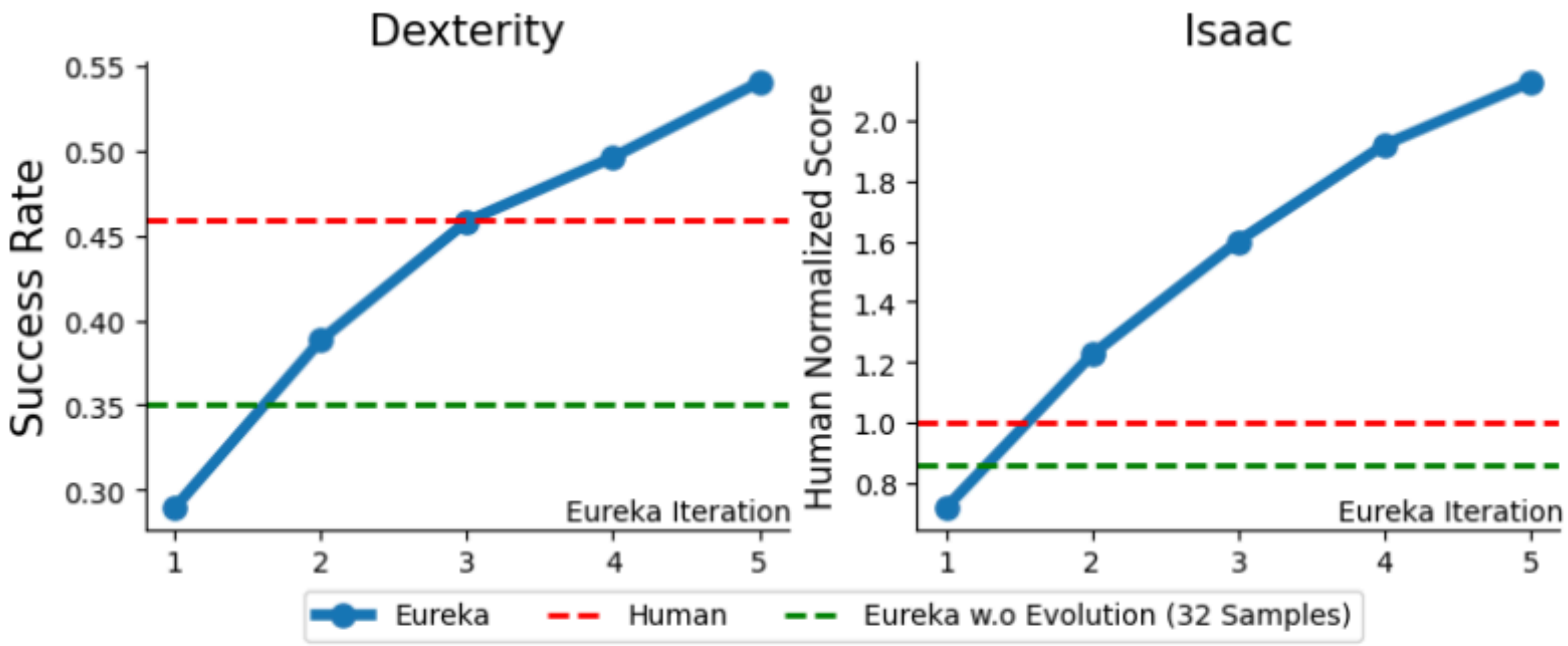



Figure 5: EUREKA progressively produces better rewards via in-context evolutionary reward search.

# Eureka

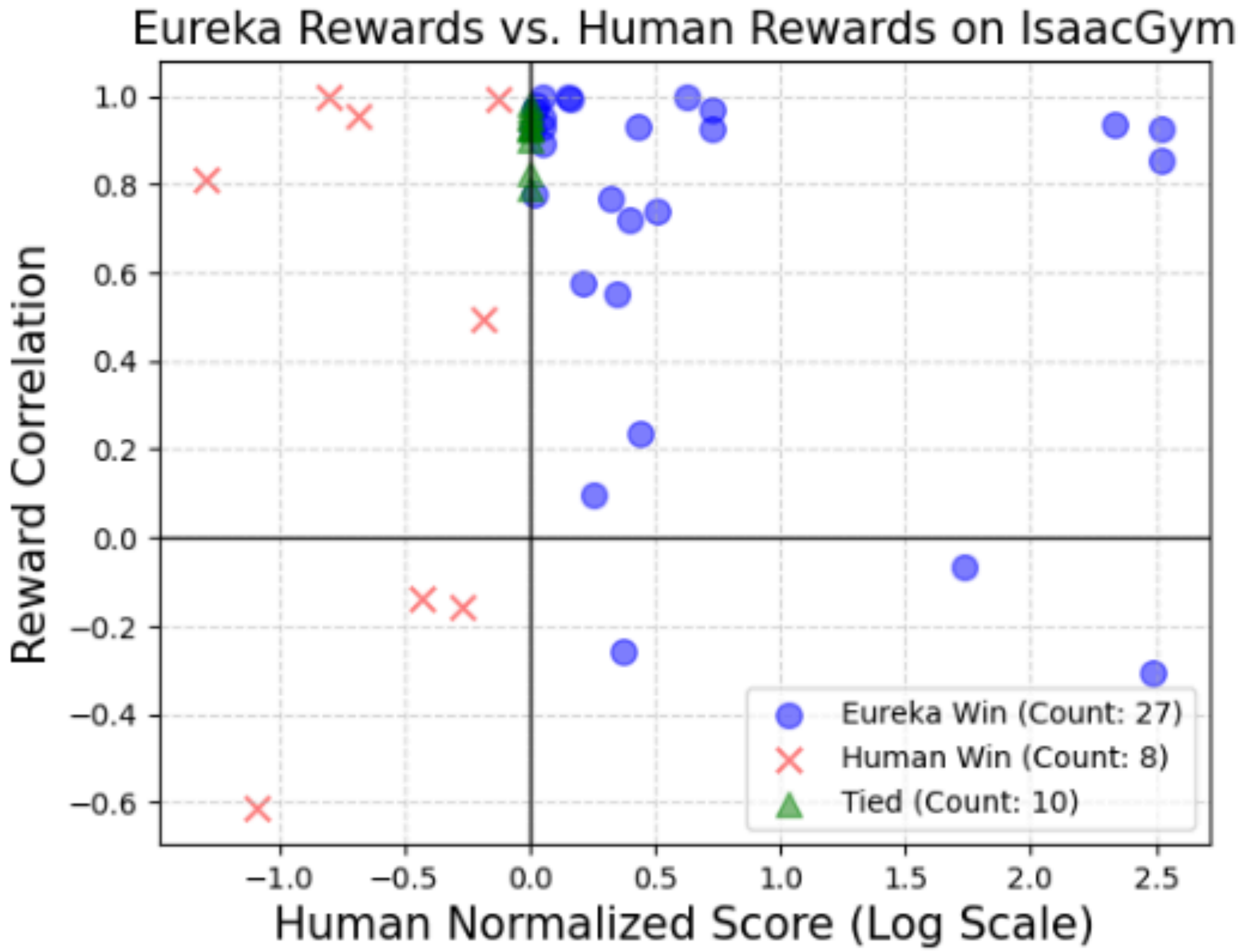



Figure 6: Eureka generates novel rewards.

# Eureka

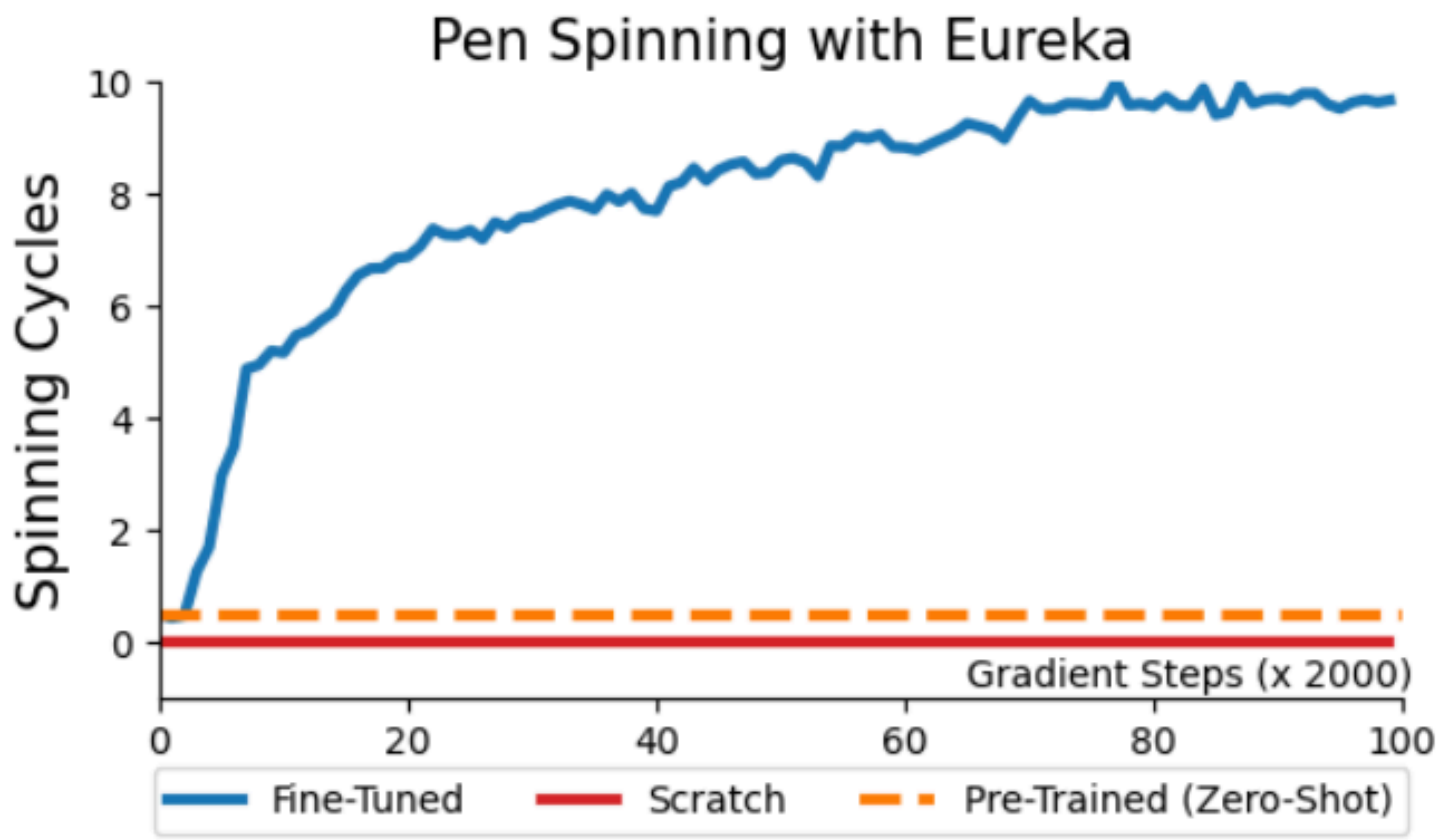



Figure 7: EUREKA can be flexibly combined with curriculum learning to acquire complex dexterous skills.

# Eureka

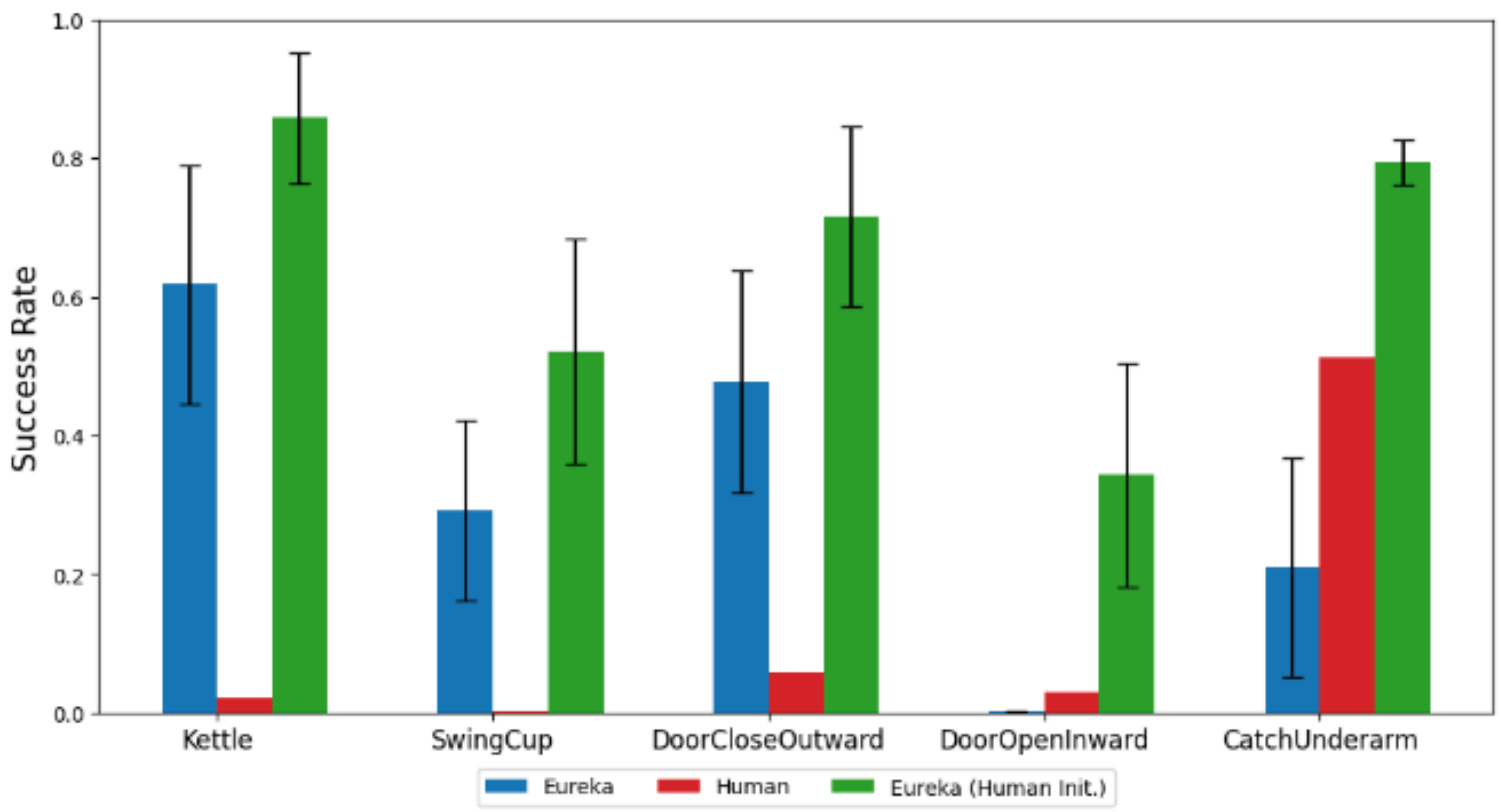



Figure 8: EUREKA effectively improves and benefits from human reward initialization.

# Eureka

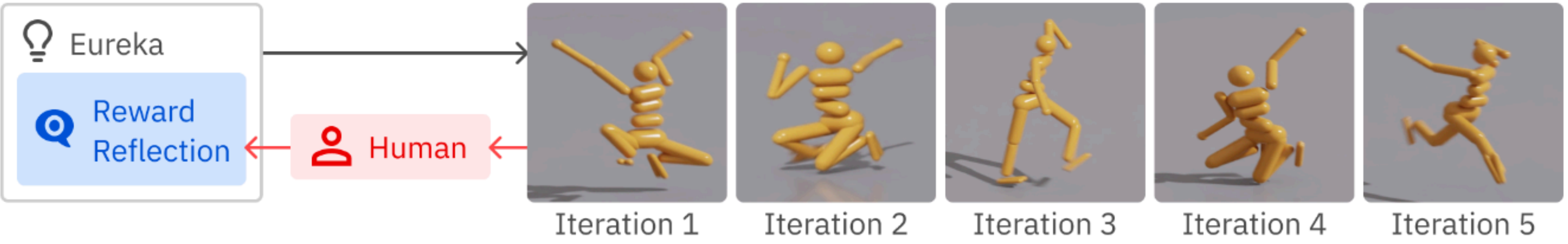



Figure 9: EUREKA can incorporate human feedback to modify rewards and induce more human-aligned policys.

| Method | Forward Velocity | Human Preference |
|---|---|---|
| EUREKA | **7.53** | 5/20 |
| EUREKA-HF | 5.58 | **15/20** |

Table 1: Human users prefer the Humanoid behavior learned via EUREKA rewards generated using human reward reflection.