Extending environments to incentivize self-reflection in reinforcement learning

Samuel Allen Alexander[1][0000−0002−7930−110X]

The U.S. Securities and Exchange Commission `samuelallenalexander@gmail.com`
https://philpeople.org/profiles/samuel-alexander/publications

Abstract. We consider an extended notion of reinforcement learning environment, in which the environment is able to simulate the agent. We give examples of some such extended environments which seem to incentivize various different types of self-reflection or self-awareness.

1 Introduction

In this paper, we consider computable agents in a deterministic variant of the reinforcement learning (RL) framework (we restrict our attention to deterministic environments and agents for the sake of simplicity, but the basic ideas would not be difficult to adapt to non-deterministic RL). Such agents are intended to interact with deterministic RL environments, where they take actions and receive rewards and observations in response to those actions.

Traditionally, a deterministic RL environment is essentially a function e which outputs a reward-observation pair $\langle r, o\rangle = e(r_0, o_0, a_0, \ldots, r_n, o_n, a_n)$ in response to an award-observation-action sequence $(a_0, \ldots, a_n)$. However, there is another type of environment in which computable RL agents can interact just as well. Essentially, we define an *extended environment* to be a function e which outputs a reward-observation pair $\langle r, o\rangle = e(T, \langle r_0, o_0, a_0, \ldots, r_n, o_n, a_n\rangle)$ in response to a reward-observation-action sequence along with a Turing machine T which computes the agent. Intuitively, this should be thought of as follows: when the agent enters the environment, the environment is made aware of the agent's source-code, and can use that source-code to simulate the agent when computing rewards and observations.

For example, imagine a game[1] consisting of a labyrinth where the player wanders from room to room, each room containing a treasure, and most (but not all) rooms containing a guard.

- The player can only see one room at a time, and cannot look ahead into adjacent rooms, nor visit the same room twice.
- In a room with no guard, the player can take the treasure, yielding a reward.
- If the player chooses to take the treasure in a guarded room, then, by simulating the player, the guard determines: "If this room had been unguarded, would the player have taken the treasure?" If so, the guard blocks the player

[1] This game bears some similarity to Newcomb's Paradox [13].

and zaps them (yielding a negative reward). Otherwise, the guard allows the player to take the treasure (and the player is rewarded).

The above game would apparently be impossible (or prohibitively expensive) for a human to play, due to the difficulty of simulating a human player. But there is no reason why the above game could not be played by an RL agent (the agent's source-code being given to the game-engine beforehand). Clearly this kind of extended environment is not the kind of environment the RL agent is intended to be applied to and is not representative of the kind of environments RL agents are applied to[2]. However, such environments could be useful on the path to Artificial General Intelligence (AGI) because they seem to incentivize self-reflection: in order to perform well across many games like the above, the player would evidently need some degree of self-awareness.

We will give examples of interesting extended environments of the above kind, designed to incentivize RL agents to recursively engage in self-reflection in various ways. We conjecture that traditional RL agents would perform poorly in these extended environments, because traditional RL techniques do not involve any sort of self-reflection. We hope these examples will facilitate new RL techniques that do involve self-reflection, hopefully as a step toward AGI.

2 Preliminaries

Definition 1. *(Plays and prompts)*

1. *By an* ROA-play*, we mean either the empty sequence $\langle\rangle$, or else a sequence of the form*
$$r_0, o_0, a_0, \ldots, r_k, o_k, a_k$$
where each $r_i \in \mathbb{Q}$ (thought of as a reward*), each $o_i \in \mathbb{N}$ (thought of as an* observation*), and each $a_k \in \mathbb{N}$ (thought of as an* action*).*
2. *By an* ROA-prompt*, we mean a sequence of the form $s \frown r \frown o$ where s is an ROA-play, $\frown$ denotes concatenation, $r \in \mathbb{Q}$ (thought of as a* reward*), and $o \in \mathbb{N}$ (thought of as an* observation*).*

Definition 2. *(Agents and environments)*

1. *An* agent *is a Turing machine which halts whenever it is run on an ROA-prompt, outputting an action $a \in \mathbb{N}$.*
2. *An* extended environment *is a Turing machine e such that:*
   (a) *For every agent T, for every ROA-play s, when e is run on input $\langle T, s\rangle$, e halts on that input, outputting a pair $\langle r, o\rangle$ where $r \in \mathbb{Q}$ (thought of as a reward) and $o \in \mathbb{N}$ (thought of as an observation).*

[2] Such environments might, however, accidentally arise if both environment and agent are implemented on the same machine and the environment is managed by an AI sophisticated enough to exploit unintended informational side channels, as in [18].

There is a subtle nuance in Definition 2. Should the agent's next action depend on the entire history (including prior actions), or only on prior rewards and observations? One could argue that the agent's next action needn't depend on its own past actions, since its own past actions can be inferred from past rewards and observations. Normally, it would not matter much whether or not the agent's next action depend on its own past actions[3]. In formalizing examples of extended environments that incentivize self-reflection, we have found it convenient for the agent's next action to formally depend on past actions. Perhaps this reflects that known conscious agents (e.g. humans) evidently do *not* carefully re-compute their own past actions from remembered observations and rewards, but instead, a human maintains memories of her past actions as well, regardless whether doing so is formally superfluous.

Definition 3. *Suppose T is an agent and e is an extended environment. The* result of T interacting with e *is the infinite reward-observation-action sequence*

$$r_0, o_0, a_0, r_1, o_1, a_1, \ldots$$

(each $r_i \in \mathbb{Q}$, $o_i, a_i \in \mathbb{N}$) defined inductively as follows.

- *r_0 and o_0 are obtained by computing e on $\langle T, \langle\rangle\rangle$.*
- *a_0 is the output of T on $\langle r_0, o_0\rangle$.*
- *For $i > 0$, r_i and o_i are obtained by computing e on*

$$\langle T, \langle r_0, o_0, a_0, \ldots, r_{i-1}, o_{i-1}, a_{i-1}\rangle\rangle.$$

- *For $i > 0$, a_i is obtained by computing T on*

$$\langle r_0, o_0, a_0, \ldots, r_{i-1}, o_{i-1}, a_{i-1}, r_i, o_i\rangle.$$

One important implication of extended environments is that they further divide the (already divided) ways of measuring intelligence of RL agents. Intelligence measures [1] [9] [12] which aggregate performance over traditional environments only measure an agent's intelligence over those environments. The same measures could easily be extended to also take extended environments into account, perhaps providing measures which better capture agents' self-reflection ability.

3 Examples of Self-reflection-incentivizing Environments

In this section, we give examples of some interesting extended environments which seem to incentivize various forms of self-reflection. We are inspired by various libraries of traditional RL environments [3] [4] [5] [7] [8]. Some of our examples do not provide individual environments, but rather, procedures for obtaining new environments from old (these could be iterated, in order to build up environments incentivizing deeper and deeper nested self-reflection).

[3] In [1], we formalize agents whose actions depend only on past rewards and observations. Legg and Hutter give a formalization where the agent's next action does explicitly depend on its past actions [12].

Example 1. (Reward Agent for Ignoring Rewards) For each ROA-prompt p, let p^0 be the ROA-prompt equal to p except that all rewards are 0. For any environment e, we define a new environment e' as follows (where T is a Turing machine, p is an ROA-prompt, and $a \in \mathbb{N}$ is thought of as the agent's action in response to p):

$$\begin{aligned} e'(T, \langle\rangle) &= e(T, \langle\rangle) \\ e'(T, p \frown a) &= \langle r, o\rangle, \end{aligned}$$

where o is the observation component of $e(T, p \frown a)$ (if e does not halt on input $\langle T, p \frown a\rangle$ (resp. $\langle T, \langle\rangle\rangle$) then neither does e') and

$$r = \begin{cases} 1 & \text{if } a = T(p^0), \\ -1 & \text{if } a \neq T(p^0) \end{cases}$$

(if T does not halt on p^0 then e' does not halt on $\langle T, p \frown a\rangle$).

In Example 1, the agent is rewarded if the agent acts the same way the agent would act if all rewards so far had been 0. Otherwise, the agent is punished. Thus, paradoxically, the agent is rewarded for ignoring rewards. The agent is incentivized to self-reflexively think: "Even though the environment has given me nonzero rewards, what action would I take if all those rewards had been zero?"

Example 2. (Reward Agent for Self-Inserting) Fix a canonical computable bijection $o \mapsto \hat{o}$ from $\mathbb{N}$ to $\mathbb{Q} \times \mathbb{N}$: thus, every observation o encodes a reward-observation pair $\hat{o} = \langle r', o'\rangle$, and every reward-observation pair is encoded by some such o. For any environment e, we define a new environment e' as follows (where T, p, a are as in Example 1, and with similar non-halting caveats as Example 1):

$$\begin{aligned} e'(T, \langle\rangle) &= e(T, \langle\rangle) \\ e'(T, p \frown a) &= \langle r, o\rangle, \end{aligned}$$

where o is such that $\hat{o} = e(T, p \frown a)$ and

$$r = \begin{cases} 1 & \text{if } a = T(p'), \\ -1 & \text{if } a \neq T(p') \end{cases}$$

where p' is the ROA-prompt obtained from p by replacing each reward-observation pair $\ldots, r_i, o_i, \ldots$ by the reward-observation pair $\ldots, \widehat{o_i}, \ldots$.

In Example 2, one might imagine e' as a room containing nothing but an arcade game e. There is nothing for the agent in the room to do except play this arcade game. When played, the arcade game visually displays rewards, but the agent merely observes them, and does not "feel" them. However, the agent is hooked up to an intravenous tube which injects pleasure into the agent when the agent *acts* as if she really feels the rewards displayed on the screen (and injects pain otherwise). In this way, the agent is incentivized to self-identify with the protagonist in the video-game, self-reflexively asking, "Which action would I take if those displayed rewards were real?"

Example 3. (Incentive to Incentivize) We define an environment e as follows (where T is a Turing machine and $r_0, o_0, a_0, \ldots, r_n, o_n, a_n$ is an ROA-play, with similar non-halting caveats as in Example 1).

$$\begin{aligned} e(T, \langle\rangle) &= \langle 0, 0\rangle \\ e(T, \langle r_0, o_0, a_0, \ldots, r_n, o_n, a_n\rangle) &= \langle r, 0\rangle, \end{aligned}$$

where

$$r = \begin{cases} 1 & \text{if } T(p') = 0, \\ 0 & \text{if } T(p') \neq 0 \end{cases}$$

where p' is the ROA-prompt $(r'_0, o'_0, a'_0, \ldots, r'_{n+1}, o'_{n+1})$ where $r'_0 = 0$, each $r'_{i+1} = a_i$, each $o'_i = 0$, and each $a'_i = T(r'_0, o'_0, a'_0, \ldots, r'_i, o'_i)$.

In Example 3, the agent observes actions, and the agent must act by choosing rewards for those observed actions. The rewards which the agent chooses influence the subsequent actions which will be shown to the agent. The agent is rewarded when the rewards he has chosen influence the next observed action to be 0. Unknown to the agent—but an intelligent self-aware agent should eventually notice the pattern—the way the observed actions are determined is by giving the agent's chosen rewards to a simulated copy of said agent. Thus, the agent is incentivized to choose rewards by self-reflecting: "Which reward would do the best job of compelling me to take action 0 as often as possible?" We might imagine the agent playing a video-game in which he sees himself in front of a keyboard. The video-game copy-agent types "100", and then a prompt appears on the screen saying, "Which reward will you give this worker for typing 100 just now?" The agent responds by choosing some reward, and sees an animation of the reward being given to the video-game copy-agent. The video-game copy-agent then types "0", and immediately the true agent is rewarded for getting the video-game copy-agent to type 0. Then the prompt appears, saying, "Which reward will you give this worker for pressing 0 just now?" And so on forever[4].

Many other interesting examples could be given. For example, an extended environment could reward agents based on how long (or how much memory) they

[4] Example 3 is interesting in that the agent, desiring the clone to take action 0 as often as possible, is incentivized to choose harsh punishments when the clone takes nonzero actions. If punishments are limited to $\mathbb{Q}$, then the agent faces a dilemma similar to one in RL cancer treatment applications. An RL doctor should be punished with an infinitely large negative reward for killing a patient, but this is impossible if rewards are restricted to real numbers [16] [19]. Similarly, the agent in Example 3 would probably like to choose infinitely large rewards, if possible, when its clone from takes action 0. This could be considered evidence in favor of generalizing RL to allow rewards from other number systems besides the real numbers, as in [2].

use to compute each action[5]. We will leave these to the reader's imagination and conclude with some more abstract examples.

Example 4. (Incentivizing Self-recognition) Let $p_0, p_1, \ldots$ be a canonical computable enumeration of all non-empty ROA-plays. We define an environment e as follows (where T, p, a are as in Example 1, and with appropriate non-halting caveats).

$$e(T, \langle\rangle) = \langle 0, p_0\rangle$$
$$e(T, p \frown a) = \langle r, p_n\rangle$$

where $n = \frac{\mathrm{len}(p\frown a)}{3}$ is the number of actions in $p \frown a$ and where

$$r = \begin{cases} 1 & \text{if } a > 0 \text{ and } a' = T(p'), \\ 1 & \text{if } a = 0 \text{ and } a' \neq T(p'), \\ 0 & \text{otherwise} \end{cases}$$

where $p_{n-1} = p' \frown a'$.

In Example 4, the agent is systematically shown all non-empty ROA-plays, and for each ROA-play, the agent either types "Looks like me" (any action > 0) or "Doesn't look like me" (0). When shown the non-empty ROA-play $p' \frown a'$, the agent is rewarded if and only if the agent correctly determines whether or not a' is the action the agent itself would take in response to p'. Thus, the agent is incentivized to self-reflect in order to ask, "If I experienced the observations and rewards in that OAR-sequence, would I act that way?"

The following example is partly motivated by [17].

Example 5. (Recognizing other aspects of oneself) All the below environments are similar to Example 4, and we describe them informally to avoid technical details.

– (Supervised learning) Assume there is a canonical, computable function f which transforms each RL agent A into a supervised learning agent $f(A)$. By a *supervised learning trial* we mean quadruple $\langle L, T, I, p\rangle$ where L is a finite set of labels, T is a sequence of images with labels from L (a *training set*), I is an unlabeled image, and $p : L \to \mathbb{Q} \cap [0, 1]$ is a function assigning to each label $\ell \in L$ a probability that ℓ is the correct label for I. We define an extended environment as follows. The agent A is sytematically shown all supervised learning trials and must take action 1 ("Looks like me") or 0 ("Doesn't look like me"), and is rewarded or punished accordingly depending whether or not $f(A)$ would output p in response to I after being trained with T.

[5] In some sense, by giving the environment access to the agent's source-code, we allow the environment to reflect the agent's own internal signals. Thus, external environments seem to generalize the idea of agents modified to manually predict their own internal signals, as in [15].

- (Unsupervised learning) Assume there is a canonical, computable function g which transforms each RL agent A into an unsupervised learning agent $g(A)$. By an *unsupervised learning trial* we mean a triple $\langle n, D, C\rangle$ where n is a positive integer, $D \subseteq \mathbb{Q}^n$ is a finite set of n-dimensional points with rational coordinates, and C is a clustering of D. We define an extended environment as follows. The agent A is systematically shown all unsupervised learning trials and must take action 1 ("Looks like me") or 0 ("Doesn't look like me"), and is rewarded or punished accordingly depending whether or not $g(A)$ would cluster D into clustering C.
- (The Turing Test) Assume there is a canonical, computable function h which transforms each RL agent A into an English-speaking chatbot $h(A)$. By a *chatbot trial* we mean a sequence of strings of English characters. We define an extended environment as follows. The agent A is systematically shown all chatbot trials and must take action 1 ("Looks like me") or 0 ("Doesn't look like me"), and is rewarded or punished accordingly depending whether or not even-numbered strings in the chatbot trial are what $h(A)$ would say in response to the user saying the odd-numbered strings.
- (Adversarial sequence prediction) Similar to the above environments, assuming a canonical computable function which transforms each RL agent into a predictor in the game of adversarial sequence prediction [10] [11].
- (Mechanical knowing agent) Assume there is a canonical, computable function i which transforms each RL agent A into a code $i(A)$ of a computably enumerable set of sentences in the language of Epistemic Arithmetic [14]; $i(A)$ is thought of as a mechanical knowing agent [6]. We define an extended environment as follows. The agent A is systematically shown all sentences in the language of Epistemic Arithmetic, with repetition, in such a way that each sentence is shown infinitely often. Upon being shown sentence ϕ for the nth time, the agent must take action 1 ("I know ϕ is true") or 0 ("I'm not sure if ϕ is true") and is rewarded if and only if ϕ is enumerated by $i(A)$ in $\leq n$ steps of computation.

The extended environments in Example 5 incentivize the agent to self-reflect, asking itself questions like "Does that look like how I would classify that image, given that training set?" or "Does that look like how I would cluster that set of points?" or "Does that conversation participant say the same things I would say?" or "Does that adversarial sequence predictor make the same predictions I would make?" or "Does that mechanical knowing agent know the same things I know?"

References

1. Alexander, S.A.: Intelligence via ultrafilters: structural properties of some intelligence comparators of deterministic legg-hutter agents. Journal of Artificial General Intelligence **10**(1), 24–45 (2019)
2. Alexander, S.A.: The Archimedean trap: Why traditional reinforcement learning will probably not yield AGI. arXiv preprint arXiv:2002.10221 (2020)

3. Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research **47**, 253–279 (2013)
4. Beyret, B., Hernández-Orallo, J., Cheke, L., Halina, M., Shanahan, M., Crosby, M.: The animal-AI environment: Training and testing animal-like artificial cognition. Preprint (2019)
5. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI gym. Preprint (2016)
6. Carlson, T.J.: Knowledge, machines, and the consistency of Reinhardt's strong mechanistic thesis. Annals of Pure and Applied Logic **105**(1-3), 51–82 (2000)
7. Chollet, F.: On the measure of intelligence. Preprint (2019)
8. Cobbe, K., Hesse, C., Hilton, J., Schulman, J.: Leveraging procedural generation to benchmark reinforcement learning. In: International conference on machine learning. pp. 2048–2056. PMLR (2020)
9. Hernández-Orallo, J., Dowe, D.L.: Measuring universal intelligence: Towards an anytime intelligence test. Artificial Intelligence **174**(18), 1508–1539 (2010)
10. Hibbard, B.: Adversarial sequence prediction. In: ICAGI. pp. 399–403 (2008)
11. Hibbard, B.: Measuring agent intelligence via hierarchies of environments. In: ICAGI. pp. 303–308 (2011)
12. Legg, S., Hutter, M.: Universal intelligence: A definition of machine intelligence. Minds and machines **17**(4), 391–444 (2007)
13. Nozick, R.: Newcomb's problem and two principles of choice. In: Rescher, N. (ed.) Essays in honor of Carl G. Hempel, pp. 114–146. Springer (1969)
14. Shapiro, S.: Epistemic and intuitionistic arithmetic. In: Studies in Logic and the Foundations of Mathematics, vol. 113, pp. 11–46. Elsevier (1985)
15. Sherstan, C., White, A., Machado, M.C., Pilarski, P.M.: Introspective agents: Confidence measures for general value functions. In: International Conference on Artificial General Intelligence. pp. 258–261. Springer (2016)
16. Wirth, C., Akrour, R., Neumann, G., Fürnkranz, J.: A survey of preference-based reinforcement learning methods. The Journal of Machine Learning Research **18**(1), 4945–4990 (2017)
17. Yampolskiy, R.V.: AI-complete, AI-hard, or AI-easy–classification of problems in AI. In: The 23rd Midwest Artificial Intelligence and Cognitive Science Conference (2012)
18. Yampolskiy, R.V.: Leakproofing singularity-artificial intelligence confinement problem. JCS **19** (2012)
19. Zhao, Y., Kosorok, M.R., Zeng, D.: Reinforcement learning design for cancer clinical trials. Statistics in medicine **28**(26), 3294–3315 (2009)