The Multi-Slot Framework: A Formal Model for Multiple, Copiable AIs

Laurent Orseau

AgroParisTech, UMR 518 MIA, F-75005 Paris, France
INRA, UMR 518 MIA, F-75005 Paris, France
laurent.orseau@agroparistech.fr

Abstract. Because advanced AI is likely in the future, so is the possibility of multiple advanced AIs. It is therefore also likely that such advanced AIs will be implemented in software that can be copied from hardware to hardware. The best existing theoretical framework for the rigorous formal treatment and prediction of such AIs are those based on the AIXI framework developed by Hutter [2]. Unfortunately, these single-agent frameworks do not allow formal treatment of multiple co-existing AIs. The current paper introduces a novel "multi-slot" framework for dealing with multiple intelligent agents, each of which can be duplicated or deleted at each step, in arbitrarily complex environments. The framework is a foundational first step in the analysis of environments that allow creation (by copying) and deletion of multiple agents. Even by focusing on the case where the agents do not interact, the notion of future of an agent is not straightforward anymore, so we propose several such definitions, leading to value functions and AIXI-like agents. Finally, the framework is shown to be sufficiently general to allow for the existence of a universal environment that can simulate all environments in parallel. A companion paper uses the multi-slot framework presented here to explore the notion of identity in man and machine.

Keywords: Universal AI, AIXI, multi-agent

1 Introduction

In the traditional agency framework [13], a single agent interacts with an environment by outputting an action and receiving an observation at each discrete time step. A reinforcement learning agent [17] can extract a reward from the observation. In this framework, the universal reinforcement learning agent AIXI is the optimal learning agent that chooses its actions so as to maximize its future expected reward [2], in all computable environments. Although this framework is very general and well sufficient for almost all practical purposes, it still relies on some strong assumptions, and we have already considered elsewhere relaxing some of these constraints, for example when agents can modify themselves [14,8], can be modified by the environment [12,9] or even be computed by it [10].

In this paper, we present a novel framework that is more general than the traditional one (but less, for simplicity, than the space-time embedded intelligence one [10]), where optimally intelligent agents can be duplicated and deleted by the environment. We call it the *multi-slot agency framework*, as agents are placed in *slots*, which can be thought of for now as robotic bodies or computer hardware. This is similar to a multi-agent framework [1] and could be used as such but, instead of focusing on the interactions between agents, which is difficult at best with universal agents [3], we provide a specialization of the framework to explicitly prevent interaction, so as to be able to provide formal results. What matters is that agents can make decisions regarding being copied or deleted.

This new framework is initially motivated by questions about personal identity, in particular the relation between information content and the hardware that supports it [11]: If a person is teleported by a device that scans and disassembles the person's body, transfers the information at the speed of light to a distant place where it entirely reassembles the same body, is it the same person? Some related thought experiments about the identity problem are treated formally in the companion paper [7], based on the framework presented here.

In the present paper, we focus on the definition of the framework and of the corresponding optimal reinforcement learning agents. However, there does not seem to be a single straightforward definition of the value functions: Indeed, in the traditional agency framework, there is no ambiguity as to what the future observations of the agent are. But what if the agent can be transfered or even copied to different locations? What or where are its future observations? How to take them into account in the definition of the optimal agents? We provide several plausible definitions of optimal agents, based either on the location of the agent or on the set of its copies. We believe this new framework is general enough to consider a wide range of new situations that are difficult or even impossible to consider in the traditional framework.

In the next section, the notation is introduced along with a quick exposure of the background on universal reinforcement learning agents. In section 3, the multi-slot framework is defined. In section 4, a few possible definitions of value functions are proposed. In section 5 we define a particular environment, that has the property of being able to simulate all computable traditional environments. Finally, we conclude with some remarks.

2 Notation and background

The paper recognizes the following notational conventions. At time step t each agent outputs action $a_t \in \mathcal{A}$ to the environment, which returns observation $o_t \in \mathcal{O}$ to the agent, from which a reward $r(o_t)$ can be extracted. An interaction history pair is denoted $h_t = a_t o_t$ with $h_t \in \mathcal{H} := \mathcal{O} \times \mathcal{A}$. The sequence of all actions up to time t is written $a_{1:t} = a_1 a_2 \dots a_t$, while the sequence $a_{1:t-1}$ is sometimes written $a_{\prec t}$, and similarly for other sequences like $h_{\prec t}$. The empty sequence is denoted λ. Tuples are notated with angle brackets, such as $\langle a, b \rangle$.

Boolean values are written $\mathcal{B} := \{0, 1\}$, where 1 signifies *true* when a truth-value is implied.

AIMU and AIXI [2]. A stochastic environment ν assigns a probability $\nu(o_{\prec t}|a_{\prec t})$ to an observation history $o_{\prec t}$ given the history of actions $a_{\prec t}$ of the agent.[1] For notational convenience, we will write $\nu(h_{\prec t}) \equiv \nu(o_{\prec t}|a_{\prec t})$, but keep in mind that environments do not assign probabilities to actions. A policy $\pi \in \Pi : \mathcal{H}^* \to \mathcal{A}$ produces an action given an interaction history: $a_t = \pi(h_{\prec t})$. The value of a policy π in an environment μ (with an optional action) is given by:

$$\begin{aligned} V^\pi_\mu(h_{\prec t}) &:= V^\pi_\mu(h_{\prec t}, \pi(h_{\prec t})) \ , \\ V^\pi_\mu(h_{\prec t}, a) &:= \sum_o \mu(o|h_{\prec t}a) \left[r(o) + \gamma V^\pi_\mu(h_{\prec t}ao) \right] \ , \end{aligned} \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor, that ensures finiteness of the value. The optimal (non-learning) agent AIMU for a given single environment μ is defined by the optimal policy $\pi^\mu(h_{\prec t}) := \arg\max_{\pi \in \Pi} V^\pi_\mu(h_{\prec t})$ (ties are broken in favor of policies that output the first lexicographical action at time t), and the optimal value is $V_\mu := V^{\pi^\mu}_\mu$. The value of a policy over a set of environments $\mathcal{M}$ (optionally after an action a) is given by:

$$V^\pi_\mathcal{M}(h_{\prec t}) := \sum_{\nu \in \mathcal{M}} w_\nu V^\pi_\mu(h_{\prec t}) \ . \tag{2}$$

Taking the prior weights of the environment as in Solomonoff's prior [16,18] $w_\nu := 2^{-K(\nu)}$ where $K(\nu)$ is the Kolmogorov complexity [6] of ν (*i.e.*, roughly, the size of the smallest program equivalent to ν on a universal Turing machine of reference), the optimal policy for a given set $\mathcal{M}$ of environments [2] is defined by $\pi^{\xi_\mathcal{M}}(h_{\prec t}) := \arg\max_{\pi \in \Pi} V^\pi_\mathcal{M}(h_{\prec t})$, and the optimal value function in $\mathcal{M}$ is $V_\mathcal{M} := V^{\pi^{\xi_\mathcal{M}}}_\mathcal{M}$. AIXI is the optimal agent on the set of all computable stochastic environments $\mathcal{M}_U$, with policy $\pi^\xi := \pi^{\xi_{\mathcal{M}_U}}$ and value function $V_\xi := V_{\mathcal{M}_U}$.

3 The multi-slot framework

In the multi-slot framework, the environment is interacting with several agents at the same time, on discrete interaction time steps. Each agent is on a given *slot*, a place where its binary program is written and computed; and can then be thought of as a computer (see Fig. 1). The number of agents can vary, and they can be copied to other slots, or deleted from their own slot, over time.

For generality, we first give a definition of a general agent in terms of self-modifying agents [8]:

Definition 1 (slot, memory space and agent). *A slot number is the index $i \in \mathcal{S} := \mathbb{N}^+$ of a memory space m^i. At any discrete time $t \in \mathbb{N}_{\geq 0}$, a memory*

[1] A stochastic environment can be seen as program in any programming language where sometimes some instructions are chosen with a probability.

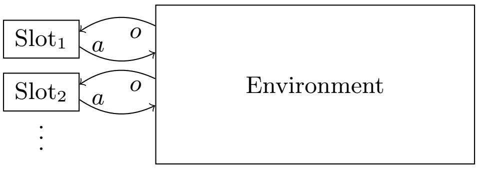

Fig. 1. The environment is in interaction with several slots.

space $m^i_t \in \mathcal{W}$ *is a bit string that can either be empty,* $m^i_t = \lambda$*, or contain an agent* $\hat{\pi}^i_t \in \hat{\Pi} : \mathcal{O} \to \mathcal{A} \times \mathcal{W}$*, i.e., a program that can be executed by an oracle computer O:* $\hat{\pi}^i_t(.) = O(m^i_t, .)$*. The agent need not be computable. An agent takes as input an observation* o_{t-1} *from the environment (or* λ *if* $t = 1$*), and generates as output (1) its current action* a_t*, and (2) a new memory* $m^i_{t^c}$ *(hence a new agent), i.e.,* $\langle a_t, m^i_{t^c} \rangle := \hat{\pi}^i_{t-1}(o_{t-1})$*, where* t^c *is the intermediate time step t right after the action is produced, named* time before copy*.*

Sequence notation $a_{1:t}$ and slot-indexed sequence notation $a^i_{1:t}$ are used for various symbols.

Definition 2 (agent set). *The agent set* $\mathcal{S}_t \in \mathcal{S}^*$ *is the set of all non-empty slots at time t:* $\mathcal{S}_t := \{i : i \in \mathcal{S}, m^i_t \neq \lambda\}$*.*

Definition 3 (copy instance, copy set, copy set list). *A copy instance* $\dot{c}^{ij}_t \in \mathcal{B}$ *is a boolean value indicating whether the memory in slot j must be copied from the memory in slot i, i.e.,* $m^j_t \leftarrow m^i_{t^c}$*. All copies are performed in parallel, yielding a new agent set. The copy set* $\dot{c}^i_t$ *is the set of all slots j at time t that are copied from slot i:* $\dot{c}^i_t := \{j : \dot{c}^{ij}_t = 1\}$*. All slots that are not in a copy set at time t are empty at time t. The copy set list is the indexed list of all non-empty copy sets at time t:* $\dot{c}_t := \{\langle i, \dot{c}^i_t \rangle : \dot{c}^i_t \neq \emptyset\}$*.*

Some derived properties:

- The copy $\dot{c}^{ii}_t$ is not implicit, *i.e.*, if $\dot{c}^{ii}_t = 0$ then m^i_t is not the copy of $m^i_{t^c}$.
- A slot j cannot be both the copy of slot i and slot k: $i \neq k \implies \dot{c}^i_t \cap \dot{c}^k_t = \emptyset$.
- The agent set $\mathcal{S}_t$ is the union of all copy sets: $\mathcal{S}_t = \bigcup_i \dot{c}^i_t$.
- Two slots being in the same copy set have the same memory content, but the converse is not always true: $j \in \dot{c}^i_t, k \in \dot{c}^i_t \Rightarrow m^j_t = m^k_t$.

Definition 4 (multi-slot interaction protocol). *The multi-slot interaction protocol between multiple, possibly non-computable agents and a single, computable environment is defined as follows. At the initial time step* $t = 0$*, the number of non-empty slots must be finite. At each time step t, the following stages occur in the given order:*

1. $t \leftarrow t+1$.

2. Each agent in a non-empty slot i produces $\tilde{\pi}^i_{t-1}(o^i_{t-1}) = \left\langle a^i_t, m^i_{t^c} \right\rangle$. The environment receives the finite set $\dot{a}_t \in (\mathcal{S} \times \mathcal{A})^*$ of pairs $\left\langle i, a^i_t \right\rangle$ of slot numbers $i \in \mathcal{S}_{t-1}$ and slot actions a^i_t for all non-empty slots i. We are now at time before copy t^c.

3. The environment computes the copy set list $\dot{c}_t$ and the copies are performed among the slots.

4. The environment outputs a finite set $\dot{o}_t \in \dot{\mathcal{O}} = (\mathcal{S} \times \mathcal{O})^*$ of pairs $\left\langle i, o^i_t \right\rangle$ of slot number $i \in \mathcal{S}_t$ and observations o^i_t for each non-empty slot i. We now have the environment interaction history triplet $\dot{h}_t := \dot{a}_t \dot{c}_t \dot{o}_t$.

5. Back to 1.

The following points are worth noting. The environment cannot know the contents of the slots, it merely performs copies. The number of non-empty slots always remains finite. The agents have no direct access to their slot numbers, to the copy sets, or to the actions performed by the other agents.

Definition 5 (multi-slot environment). *A multi-slot environment operates according to the interaction protocol in Definition 4 and is defined by a probability distribution $\dot{\nu}(\dot{c}_{1:t}\dot{o}_{1:t}|\dot{a}_{1:t})$ over all observation sequences and copy set list sequences given action sequences.*

Definition 6 (slot history). *A slot history $s_{t_1:t_2} \in \mathcal{S}^*$ is a sequence of slots resulting from a given sequence of chained copy instances $\dot{c}^{ab}_{t_1+1}\dot{c}^{bc}_{t_1+2}\dots\dot{c}^{wx}_{t_2-1}\dot{c}^{xy}_{t_2}$ such that $\forall k \in [t_1+1, t_2], \dot{c}^{ij}_k \Leftrightarrow s_{k-1} = i \wedge s_k = j$.*

In other words, the slot history $s_{9:13} = (8, 2, 3, 3, 7)$ corresponds to the sequence of copies $\dot{c}^{8,2}_{10}\dot{c}^{2,3}_{11}\dot{c}^{3,3}_{12}\dot{c}^{3,7}_{13}$.

At time t, the contents of each slot i has undergone a unique slot history $s^i_{0:t}$ of chained copies from time 0 to t, where $s^i_t = i$. The sequence is unique since a slot j at t can have been copied from only one slot i at $t-1$ (see Definition 3). For an agent in slot i at time t, this is the sequence of slots on which the actions and observations of its interaction history have been output and received.

Definition 7 (agent/environment slot action/observation/interaction history). *An environment slot interaction history $\dot{h}^i_{t_1:t_2}$ is the set of observations and actions occurring between t_1 to t_2 following the slots of the slot history $s^i_{t_1-1:t_2}$:*

$$\dot{h}^i_{t_1:t_2} := \dot{a}^{s_{t_1-1}}_{t_1} \dot{c}^{s_{t_1-1}s_{t_1}}_{t_1} \dot{o}^{s_{t_1}}_{t_1} \dots \dot{a}^{s_{t_2-1}}_{t_2} \dot{c}^{s_{t_2-1}s_{t_2}}_{t_2} \dot{o}^{s_{t_2}}_{t_2} \ .$$

Likewise with the environment slot observation history $\dot{o}^i_{t_1:t_2}$ and action history $\dot{a}^i_{t_1:t_2}$, and the agent slot observation history $o^i_{t_1:t_2}$, action history $a^i_{t_1:t_2}$, and interaction history $h^i_{t_1:t_2} = o^j_{t_1}a^j_{t_1}\dots o^i_{t_2}a^i_{t_2}$.

Thus, $\dot{h}^i_{1:t}$ is the history of actions, observations, and (chained) copies that resulted in the contents of slot i at step t from the point of view of the environment (*i.e.*, accompanied by slot numbers and copy sets), whereas $h^i_{1:t}$ is the same history but from the agent's point of view (*i.e.*, without slot numbers or copy sets).

Definition 8 (history-based agent). *On slot i at time t^c before the copies, a history-based agent is an agent whose memory content is composed of an interaction history $h^i_{\prec t}a^i_t$, and a policy π^i_t that uses the history to output an action:*

$$\textit{if } m^i_{t-1} \equiv \left\langle h^i_{\prec t}, \pi^i_{t-1} \right\rangle \textit{ then } m^i_{t^c} \equiv \left\langle h^i_{\prec t}a^i_t, \pi^i_{t-1} \right\rangle \ , \qquad a^i_t := \pi^i_{t-1}(h^i_{\prec t}) \ .$$

We define the probability of an agent interaction history as $\dot\nu(h^i_{\prec t}) := \sum_{\dot h_{\prec t}} \dot\nu(\dot h_{\prec t})\dot\nu(h^i_{\prec t}|\dot h_{\prec t})$ where $\dot\nu(h^i_{\prec t}|\dot h_{\prec t})$ is either 1 or 0 depending on whether $h^i_{\prec t}$ is consistent with $\dot h_{\prec t}$.

Definition 9 (history-based multi-slot environment). *A history-based multi-slot environment is a multi-slot environment $\dot\nu$ so that the copy set of a given slot i and the observations output to these copies depend only on the history of the agent in slot i and the involved slot numbers:*

$$\forall j : \dot\nu(\dot c^{ij}_t \dot o^j_t | \dot h_{\prec t}\dot a_t) = \dot\nu(\dot c^{ij}_t \dot o^j_t | h^i_{\prec t}a^i_t, i, j) \ .$$

These environments ensure, as a simplification, that there is no interaction between the different agents. Note however that the copy instances and observations can depend on the slot number, which means that an agent in slot i may not have the same interaction with the environment than an agent in slot j with the same history.

Note that even though the agents cannot interact, the "sub-environments" they are interacting with are still not independent, because they are all the continuations of the environments at the previous step and, since initially we will consider that there is only one agent in slot 1, all the sub-environments are tied to this initial environment. Furthermore, because the sub-environments are generated by a single program, the more different the sub-environments, the higher the complexity of the history-based multi-slot environment.

4 Value functions for multi-slot environments

For a given mono-slot environment, AIMU is the optimal agent that chooses its actions so as to maximize its future rewards, which are extracted from its observations. But when a agent can be copied at any step, what constitutes the *future* of this agent, *i.e.*, what will be its future observations?

Interestingly, there does not seem to be any single obvious direct translation of AIMU in the multi-slot framework. For example, all the following definitions of future (observations) are valid. This then leads to various definitions of AIMU. Assuming an AIMU agent is in slot i at time t, to compute its reward for time $t+1$ it can take into account for the next time step (and thus, recursively, for all future time steps):

- only the observations output to slot 1;
- only the observations output to slot i;
- only the observations output to the tth prime number slot;

- only the observations received by the first of its copies;
- observations received by all of its copies, possibly with some weighting;
- only the observations received by its copies that yield a minimal reward;
- observations received by all agents that have the exact same history, independently of whether they have a common ancestor with the agent under consideration;
- observations received by all agents that have output the sequence 11011 of actions at some point in their history;
- observations received by all agents that have a common ancestor with the agent under consideration;
- etc.

In the following subsections, we focus on 3 AIMU agents: The *copy-centered agent* AIMU^{cpy}, which considers the observations of all of its copies; and the *static slot-centered agent* AIMU^{sta} and *dynamic slot-centered agent* AIMU^{dyn}, which consider only the observations on either a predefined slot or on the slot they think they occupy.

We assume that there is only one agent in slot 1 at $t = 0$, *i.e.*, $\mathcal{S}_0 = \{1\}$, in interaction with a history-based multi-slot environment $\dot{\nu}$.

A note on time-consistency. When designing an AIMU equation (Bellman optimality equation), such as the ones that follow, a particular attention must be paid to *time consistency* [5]: The action that the agent predicts it will take in a future situation must be the same as the one it takes if the situation actually arises. In other words, its behavior must be consistent over time with the behavior it has predicted it will have.

4.1 Estimating the current slot number

When the environment makes a copy of the agent at time t to slots i and j, this leads to two identical agents at time $t+1$ with the same history, if they receive the same observation, and the two agents therefore cannot know for sure if they are in slot i or in slot j, and must estimate the probability of being in each, based on their interaction history $h_{\prec t}$. This estimation is important as the received observations can depend on the slot number. It is also related to the need to account for the observer's localization when estimating the posterior probability of being in a given environment [4].

Then the probability that the agent is in slot i, given its interaction history $h_{\prec t}$ is defined by:

$$P^i_{\dot{\nu}}(h_{\prec t}) := \frac{\dot{\nu}(h^i_{\prec t} = h_{\prec t})}{\sum_j \dot{\nu}(h^j_{\prec t} = h_{\prec t})}$$

and is defined as 0 if the denominator is null. In a deterministic environment, $\dot{\nu}(h^i_{\prec t} = h_{\prec t})$ is either 1 or 0 in depending on whether the history $h^i_{\prec t}$ is consistent with the environment, and the denominator is the number of agents that have the same history.

4.2 The copy-centered agent AIMU$^{\text{cpy}}$

The copy-centered agent AIMU$^{\text{cpy}}$ considers that its future observations are the observations received by all of its copies. Unfortunately, it is not possible to simply assign a weight of 1 to each copy. For example, if the number of copies grows faster than the geometric discounting decreases, the value function may not be summable, which prevents the comparison of action values in general. Therefore, it is necessary to weight the copies so as to ensure finiteness of the value function. We arbitrarily choose a uniform weighting of the copies of the next time step. Thus, at some time step t, if the agent is in slot i, the weight of one of its direct copies in slot j is defined by

$$P^{ij}_{\dot\nu}(h_{\prec t}a) := \frac{\dot\nu(\dot c^{ij}_t | h^i_{\prec t}a^i_t = h_{\prec t}a)}{\sum_k \dot\nu(\dot c^{ik}_t | h^i_{\prec t}a^i_t = h_{\prec t}a)}$$

and is defined as 0 if the numerator is 0. In deterministic environments, this is simply the inverse of the size of the copy set, $\frac{1}{|\dot c^i_t|}$.

We thus define the copy-centered agent value function as:

$$V^{\text{cpy}}_{\pi,\dot\mu}(h_{\prec t}, a) := \underbrace{\sum_i P^i_{\dot\mu}(h_{\prec t})}_{\substack{\text{Estimate the}\\ \text{current slot}}} \underbrace{\sum_j P^{ij}_{\dot\mu}(h_{\prec t}a)}_{\text{Weight each copy}} \underbrace{\sum_{o^j_t} \dot\mu(\dot c^{ij}_t o^j_t | h^i_{\prec t}a^i_t = h_{\prec t}a_t)}_{\substack{\text{Take action and receive}\\ \text{observation in copy slot}}} \Big[r(o^j_t) + \gamma V^{\text{cpy}}_{\pi,\dot\mu}(h_{\prec t}ao^j_t)\Big] .$$

Note that the recurrence follows the generic definition of Equation (1). We can simplify the equation by defining:

$$\hat\mu(\dot c^{ij}_t o^j_t | h_{\prec t}a_t) := P^i_{\dot\mu}(h_{\prec t}) P^{ij}_{\dot\mu}(h_{\prec t}a_t) \dot\mu(\dot c^{ij}_t o^j_t | h^i_{\prec t}a^i_t = h_{\prec t}a_t)$$

to have:

$$V^{\text{cpy}}_{\pi,\dot\mu}(h_{\prec t}, a) := \sum_{i,j,o^j_t} \hat\mu(\dot c^{ij}_t o^j_t | h_{\prec t}a) \Big[r(o^j_t) + \gamma V^{\text{cpy}}_{\pi,\dot\mu}(h_{\prec t}ao^j_t)\Big] .$$

Following the generic definitions of section 2, and as for the following value functions, the value of the optimal policy is $V^{\text{cpy}}_{\dot\mu}$, which defines the optimal non-learning agent AIMU$^{\text{cpy}}$, the value of the optimal policy in a set $\dot{\mathcal{M}}$ is $V^{\text{cpy}}_{\dot{\mathcal{M}}}$, and the optimal learning agent for all computable stochastic history-based multi-slot environments $\dot{\mathcal{M}}_U$ is AIXI$^{\text{cpy}}$ with value $V^{\text{cpy}}_{\dot\xi}$.

Some remarks:

- Once the slot number is estimated, one may expect this information to be passed through the recurrence on $V^{\text{cpy}}_{\dot\mu}$. However, doing so would not be time-consistent: Indeed, the agent at time $t+1$ will not have access to this information, and therefore the agent at time t must do as if it were the agent at $t+1$, even though it already has estimated the slot number. (Thus $V^{\text{cpy}}_{\pi,\dot\mu}(h_{\prec t}ao^j_t)$ must be read $V^{\text{cpy}}_{\pi,\dot\mu}(h_{\prec t}ao)$.)

- $\dot{c}^{ij}_t$ must be kept inside $\dot{\mu}(\dot{c}^{ij}_t o^j_t|\ldots)$ despite the weight of the copies. For example, if there is probability 0.5 for the agent to be copied to only 1 slot (thus half of the time resulting in no copy), the weight of this copy is still 1 (whereas the copy probability is 0.5).

4.3 The static slot-centered agent AIMU$^{\text{sta}}$

The static slot-centered agent AIMU$^{\text{sta}}$ considers only the observations received on a particular predefined slot. As an analogy, this slot can be seen as a bank account, on which the agent wants to maximize the value, but does not care about the value on any other account. The static slot-centered agent AIMU$^{\text{sta}}$ value function is defined by:

$$V^{\text{sta},i}_{\pi,\dot{\mu}}(h_{\prec t}, a) := \sum_{o^i_t} \dot{\mu}\left(\dot{c}^{ii}_t o^i_t | h^i_{\prec t} a^i_t = h_{\prec t} a_t, \dot{c}^{ii}_{\prec t}\right) \left[r(o^i_t) + \gamma V^{\text{sta},i}_{\pi,\dot{\mu}}(h_{\prec t} a o^i_t)\right] \ ,$$

with $\dot{\mu}(\cdot|x) := 0$ if $\dot{\mu}(x) = 0$.

This agent cares only about what happens on a predefined slot, and all its copies will also care about the same slot. Furthermore, the arguable presence of $\dot{c}^{ii}_t$ and $\dot{c}^{ii}_{\prec t}$ requires that the agent "stays" on slot i and is not moved from one slot to another before coming back to the initial slot, and for time consistency, consider only cases where it has always been on slot i in the past.[2]

4.4 The dynamic slot-centered agent AIMU$^{\text{dyn}}$

A more dynamic version of the slot-centered agent takes into account the observation (for the near and far future) of the (estimated) slot it occupies at the current time step.

As it is not told what slot it should care about, it must estimate it given its interaction history. But this leads to a complication: Say at time t the agent knows it is in slot i, and will be copied to slots i and j at time $t{+}1$, independently of the action, and both future agents will receive the same observation. The agent in slot i at t should want to optimize what happens in slot i only, but there is a trick: the agent at time $t+1$ in slot i will not have (in general) the information of its slot being i and not j, and therefore must estimate it—it will thus estimate that it can be either in slot i or in slot j. Thus, the agent at time t must optimize what will happen on slot i only, knowing that its copy at $t{+}1$ will optimize what happens on both slots i and j. This time consistency requirement leads to the

[2] However, removing these would require the agent to consider the actions of other agents that could come on its slot, which is an open problem [3].

following value function for the dynamic slot-centered agent AIMU$^{\text{dyn}}$:

$$V^{\text{dyn}}_{\pi,\dot\mu}(h_{\prec t}, a) := \underbrace{\sum_i P^i_{\dot\mu}(h_{\prec t})}_{\text{estimate current slot}} \underbrace{V^{\text{dyn},i}_{\pi,\dot\mu}(h_{\prec t}, a)}_{\text{value on slot } i},$$

$$V^{\text{dyn},i}_{\pi,\dot\mu}(h_{\prec t}, a) := \sum_{o^i_t} \dot\mu(\dot c^{ii}_t o^i_t | h^i_{\prec t} a^i_t = h_{\prec t} a) \Bigg[r(o^i_t) + \gamma \underbrace{V^{\text{dyn},i}_{\pi,\dot\mu}\Big(h_{\prec t} a o^i_t, \underbrace{V^{\text{dyn}}_{\pi,\dot\mu}(h_{\prec t} a o^i_t)}_{\text{behavior of the future agent}} \Big)}_{\text{value on slot } i \text{ of the behavior of the future agent}} \Bigg].$$

The value functions and their corresponding optimal agents defined above are all plausible transformations of AIMU and AIXI for multi-slot environments, but it remains to understand if they really behave correctly in all situations (the very definition of "correctly" being the problem itself). Some experiments are set up in the companion paper [7] to provide first insights.

5 The universal multi-slot environment

In history-based environments, when a agent is copied to two slots, as the two new agents do not interact anymore, they can be seen as being in parallel (mono-slot) universes. We can even construct a very simple "universal" environment that simulates all the mono-slot environments in parallel.

Let $\dot\mu^\forall$ be the multi-slot environment defined as follows: At each time step, each agent in slot i is copied to slot i and to slot $2^{t-1}+i$ (thus $\dot c^i_t = \{i, 2^{t-1}+i\}$). The first copy receives observation 0 and the second one observation 1.

$$\forall t, i \leq 2^{t-1} : \dot\mu^\forall(\dot c^{ij}_t o^j_t) = \begin{cases} 1 & \text{if } j = i \text{ and } o^j_t = 0 \\ 1 & \text{if } j = 2^{t-1}+i \text{ and } o^j_t = 1 \\ 0 & \text{otherwise} \end{cases}$$

We call this environment universal because of the following theorem:

Theorem 1. *In the universal environment $\dot\mu^\forall$, with $\mathcal{O} = \{0,1\}$, for any deterministic mono-slot environment $q \in \mathcal{Q}$, at any time $t > 0$, there is always one slot $i \leq 2^t$ (and exactly one) so that the interaction history $h^i_{1:t} \equiv (a_{1:t}, o_{1:t})$ of the agent in this slot is consistent with q, i.e., $q(a_{1:k}) = o_k$ $\forall k, 0 < k \leq t$.*

Proof. By recurrence. For such a given environment q, suppose there is a slot i so that the current interaction history $h^i_{\prec t} \equiv (a_{\prec t}, o_{\prec t})$ on this slot is consistent with q: $q(a_{\prec t}) = o_k$ $\forall k, 0 < k < t$. Let $a_t = a^i_t$ be the action chosen by the agent on this slot i. Then the history on slot i at the end of time step t is $h_{\prec t}a_t0$ and the history on slot $j = 2^{t-1}+i$ is $h_{\prec t}a_t1$. As $q(h_{\prec t}a_t)$ is either 0 or 1, the history on either slot j or slot i (but not both) is still consistent with q. As the empty history at $t = 0$ is consistent with all environments, the recurrence holds. □

Note that, interestingly, this theorem would not hold if the observations were given to the agent in slot i before the copies are performed, because then the histories on slots i and j would be the same.

The universal environment is reminiscent of the coin-flip stochastic environment, that outputs observation 0 or 1 with a $\frac{1}{2}$ probability, but there is an important difference: In the stochastic environment, any observation string of length t has a probability 2^{-t} to be observed, whereas in the universal environment, *all* strings of length t are always realized, which means that the probability that an agent has observed the string 0^t is 1, just like there is at least another agent at time t (sufficiently big) that has an interaction history corresponding to playing chess games in a mono-slot environment (also compare Schmidhuber's multiverse [15]).

Since the interaction history $h_{\prec t}$ is unique on each slot, for each interaction history there exists a single slot i so that $P^i_{\dot{\mu}^\forall}(h_{\prec t}) = 1$. Therefore, it implies that the universal environment has a constant weight in the value function, but it has no predictive power as observations are the same for all actions and thus does not bias the selection of the action.[3]

6 Conclusion

The multi-slot framework allows formal treatment of multiple, simultaneous intelligent agents that can be duplicated or deleted. Each agent inhabits its own slot and can be copied to another slot at each time step. Restricting attention to history-based environments ensures that the agents do not interact, thereby allowing the definition of optimal, incomputable agents, such as Hutter's AIMU and AIXI [2].

The new framework opens up a broad range of definitions for such agents, leading to several different definitions of value function corresponding to very different ways of valuing the future. The copy-centered agent, for example, plans the future of each of its copies with equal weight, while the slot-centered agent attempts only to optimize the future of a particular slot.

The framework is sufficiently general to allow the existence of a universal environment, which simulates all other environments in parallel. This universal environment has a constant probability of being the true environment at all steps, which raises some epistemological questions regarding what is truly knowable about our world.

In a companion paper [7], the multi-slot framework provides the foundation for several thought experiments, allowing formal results regarding the nature of personal identity (natural and artificial).

[3] But note that a similar environment that exchanges the observations depending on the action $a \in \{0, 1\}$ would have some predictive power.

Acknowledgements

Thanks especially to Mark Ring for help on earlier drafts and for our many extensive discussions, from which this paper arose, regarding the nature of identity. Thanks also to Stanislas Sochacki for earlier formative conversations on this topic, and to Jan Leike for helpful comments and careful reading.

References

1. Ferber, J.: Multi-agent systems: an introduction to distributed artificial intelligence, vol. 1. Addison-Wesley Reading (1999)
2. Hutter, M.: Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability. Springer (2005)
3. Hutter, M.: Open problems in universal induction & intelligence. Algorithms 3(2), 879–906 (2009)
4. Hutter, M.: Observer localization in multiverse theories. In: Proceedings of the Conference in Honour of Murray Gell-Mann's 80th Birthday. pp. 638–645. World Scientific (2010)
5. Lattimore, T., Hutter, M.: Time Consistent Discounting. In: Algorithmic Learning Theory. LNAI, vol. 6925, pp. 383–397. Springer, Berlin, Espoo, Finland (2011)
6. Li, M., Vitanyi, P.: An Introduction to Kolmogorov Complexity and Its Applications. Springer-Verlag, third edit edn. (2008)
7. Orseau, L.: Teleporting universal intelligent agents. In: Artificial General Intelligence (AGI). LNAI, vol. 8598. Springer (2014)
8. Orseau, L., Ring, M.: Self-Modification and Mortality in Artificial Agents. In: Artificial General Intelligence (AGI). pp. 1–10. LNAI, Springer (2011)
9. Orseau, L., Ring, M.: Memory issues of intelligent agents. In: Artificial General Intelligence, Lecture Notes in Computer Science, vol. 7716, pp. 219–231. Springer Berlin Heidelberg (2012)
10. Orseau, L., Ring, M.: Space-time embedded intelligence. In: Artificial General Intelligence, Lecture Notes in Computer Science, vol. 7716, pp. 209–218. Springer Berlin Heidelberg (2012)
11. Parfit, D.: Reasons and Persons. Oxford University Press, USA (1984)
12. Ring, M., Orseau, L.: Delusion, Survival, and Intelligent Agents. In: Artificial General Intelligence (AGI). pp. 11–20. LNAI, Springer (2011)
13. Russell, S.J., Norvig, P.: Artificial Intelligence. A Modern Approach. Prentice-Hall, 3rd edn. (2010)
14. Schmidhuber, J.: Ultimate cognition à la Gödel. Cognitive Computation 1(2), 177–193 (2009)
15. Schmidhuber, J.: The fastest way of computing all universes. In: A Computable Universe: Understanding and Exploring Nature as Computation, pp. 381–398. World Scientific (2012)
16. Solomonoff, R.: Complexity-based induction systems: comparisons and convergence theorems. IEEE transactions on Information Theory 24(4), 422–432 (1978)
17. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press (1998)
18. Zvonkin, A K and Levin, L.A.: The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. Russian Mathematical Surveys 25(6), 83–124 (1970)