Tiling agents in causal graphs

Nate Soares

May 2014

Fallenstein and Soares [2014] demonstrates that it's possible for certain types of proof-based agents to "tile" (license the construction of successor agents similar to themselves while avoiding Gödelian diagonalization issues) in environments about which the agent can prove some basic nice properties. In this technical report, we show via a similar proof that causal graphs (with a specific structure) are one such environment. We translate the proof given by Fallenstein and Soares 2014 into the language of causal graphs, and we do this in such a way as to simplify the conditions under which a tiling meliorizer can be constructed.

What does it mean for an agent to construct another agent in a causal graph? For that matter, what does it mean for an agent to be in a causal graph? When we talk about an *agent* in a causal graph, we are discussing a node that has some set of *input nodes* and some set of *output nodes*, such that the agent node's contribution to each output node is defined by some deterministic function (the agent) of the input nodes.

In figure 1, the agent is at the node **A** with observation nodes **O**... and output nodes **X**.... The agent itself is the function $\pi : \mathbf{O}\ldots \to \mathbf{X}\ldots$ that defines the agent's contribution to each output node given each input node. Without loss of generality, we can assume in this paper that there is only one observation node **O**.

We say that an agent can *construct a successor* if one of the agent's output nodes is another agent node. An agent **A** can construct a successor at the node **A**′ if the agent's contribution to **A**′ can make **A**′ be another agent, as in figure 2. This notion will be formalized in the following sections.

Following Fallenstein and Soares [2014], we consider *proof based agents*, which may only take actions after exhibiting a proof of certain safety properties. Specifically, we consider agents using the *suggester-verifier* architecture, which contain both a *suggester* component that searches for action/proof pairs and a *verifier* component that takes an action/proof pair and executes the action if and only if the given proof is a valid proof proving the appropriate safety properties.

We limit our consideration to *meliorizing* suggester-verifier agents, which possess a *fallback policy* to execute if no valid action/proof pair is found. The target theorem (in the sense of Schmidhuber [2007]) for these agents is one stating that executing the given action results in more expected utility than executing the fallback policy. Thus, in order to execute an action a, the suggester must find both the action a and a proof (in the language of the verifier) that executing a results in more expected utility than executing the fallback policy.

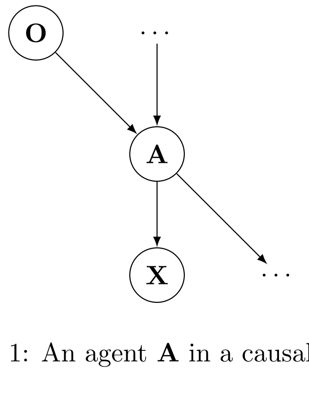

Figure 1: An agent **A** in a causal graph

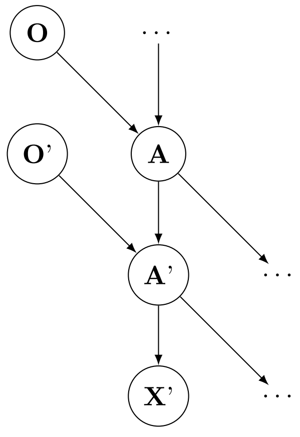

Figure 2: An agent $\mathbf{A}$ that can potentially create another agent $\mathbf{A}'$

Due to Gödelian reflection issues, it is somewhat difficult for a meliorizing suggester-verifier agent to construct another meliorizing suggester-verifier agent (as the former must trust the soundness of the latter's verifier). This act (an agent verifying the construction of a similar successor agent) is referred to as *tiling*.

The following argument demonstrates that a meliorizing suggester-verifier agent can tile in a causal graph if it is sufficiently confident that the graph has a structure like in 2. This fact is proven below.

1 Tiling when the graph is provably programmable

We consider an agent inhabiting node $\mathbf{A}$ in a causal graph. It has observation node $\mathbf{O}$, which is a random variable over the finite set $\mathcal{O}$ of possible observations that the agent can make. The agent has available to it a finite set $\mathcal{A}$ of actions, all affecting its output nodes. The agent itself is defined by a function $\pi : \mathcal{O} \to \mathcal{A}$. In this technical report, we consider meliorizing suggester-verifier agents.

A meliorizing suggester-verifier agent is an agent with a fallback policy π_{fb} and a utility function $\mathcal{U}$, and which uses a suggester component σ to search for action/proof pairs (a, p) such that p is a valid proof (in some theory $\mathcal{T}$) that the expected utility of executing a exceeds the expected utility of executing π_{fb}.

Thus, when we say that an agent π is a meliorizing suggester-verifier agent, we are claiming that there exists some theory $\mathcal{T}$, utility function $\mathcal{U}$, suggester σ, and fallback policy π_{fb} such that $\pi = \text{sv}_{\mathcal{T}}(\ulcorner \mathcal{U} \urcorner, \sigma, \pi_{\text{fb}})$ where $\text{sv}_{\cdot}(\cdot, \cdot, \cdot)$ is a function implementing the suggester-verifier behavior described above, with details as given by Fallenstein and Soares [2014].

Specifically, in order to license the execution of the action a after making the observation o, the suggester component σ must find a proof p of the *target theorem* given by (1), where $\overline{o}$ is the numeral denoting the metalanguage variable o and $\overline{a}$ is the numeral denoting the metalanguage variable a.

$$\ulcorner \mathbb{E}[U \mid \mathbf{O} = \overline{o}, \text{do}(\mathbf{A} := \overline{a})] \geq \mathbb{E}[U \mid \overline{o}, \text{do}(\mathbf{A} := \pi_{\text{fb}}(\overline{o}))] \urcorner \tag{1}$$

When we say that the agent is considering the construction of another agent, we mean that among the agent's output nodes is a node $\mathbf{A}'$ with an observation node $\mathbf{O}'$ random over the finite set of possible child observations $\mathcal{O}'$ such that $\mathbf{A}$ may cause $\mathbf{A}'$ to be a specific function $\pi' : \mathcal{O}' \to \mathcal{A}'$ (where $\mathcal{A}'$ is the finite set of actions available to affect the output nodes of $\mathbf{A}'$).

We are concerned with whether a meliorizing suggester-verifier agent *capable* of creating another agent can *license* the creation of another agent. The agent $\mathbf{A}$ may have available many actions a that program the node $\mathbf{A}'$ with some policy π', but we are concerned specifically with actions a that program $\mathbf{A}'$ such that it acts similarly to $\mathbf{A}$. If some such actions a can be licensed (e.g., if there exists a proof p of (1) about a) then $\mathbf{A}$ may be capable of self-modification. Intuitively, this may seem rather trivial, but in fact, such proofs are hindered by Gödelian issues of self-reference, as discussed by Fallenstein and Soares [2014].

In general, a meliorizing suggester-verifier agent $\mathbf{A}$ should not be able to license the implementation of *any* policy π' in the child node $\mathbf{A}'$, as many such policies will not in fact be safe. Even among safe policies π', only some of those policies are *provably* safe. In this technical report, we show that there is a subset of policies π' that are both "provably safe", ((1) can be proven about actions implementing π') and which are reasonably similar to the policy π implemented by the parent agent $\mathbf{A}$.

Specifically, we consider policies π' which, like the parent policy, implement meliorizing suggester-verifiers. We restrict consideration to suggester-verifiers π' with a fallback policy that is at least as good as the fallback policy of $\mathbf{A}$, and such that the verifier component used by π' is sound.

It is not enough for these properties to be true of the child agent: in order for a proof p of (1) to exist, these properties must be *provable* in the theory $\mathcal{T}$ used by $\mathbf{A}$. *Given* that these properties are provable, we will discuss conditions under which an agent $\mathbf{A}$ inhabiting a causal graph can license an action a implementing π' in the node $\mathbf{A}'$. We show that if the following statements about π' and about the structure of the causal graph hold true, the meliorizing suggester-verifier $\mathbf{A}$ can license the implementation of π' in $\mathbf{A}'$.

The following lemmas all assume that the agent $\mathbf{A}$ has made some observation o and is considering some action a which implements the policy π' (this claim is made precise later). Much of the reasoning done by $\mathbf{A}$ is done conditioned upon $\ulcorner \mathbf{O} = \overline{o} \urcorner$ and under the counterfactual $\ulcorner \mathrm{do}(\mathbf{A} := \overline{a}) \urcorner$. For clarity, we define an abuse of notation $\mathbb{E}_{oa}[\,\cdot\,|\,\cdot\,] := \mathbb{E}[\,\cdot\,|\mathbf{O} = \overline{o}, \mathrm{do}(\mathbf{A} := \overline{a}), \,\cdot\,]$. Now, we show that in order for the agent $\mathbf{A}$ to license a implementing a suggester-verifier π' it is sufficient for the following lemmas to be provable in T:

First, the *suggester verifier* lemma, which roughly states that π' must provably implement the suggester-verifier architecture. For this, we require that for some child policy $\pi'_{\mathrm{fb}} : \mathcal{O}' \to \mathcal{A}'$ and child verifier theory $\mathcal{T}'$,

$$\begin{aligned} \mathcal{T} \;\vdash\; & \forall o' \in \mathcal{O}', a' \in \mathcal{A}'.\; (\pi'(o') = a') \to \\ & \quad (a' = \pi'_{\mathrm{fb}}(o')) \;\vee \Box_{\mathcal{T}'} \ulcorner \mathbb{E}_{oa}[U|\mathbf{O}' = \overline{o'}, \mathrm{do}(\mathbf{A}' := \overline{a'})] \geq \mathbb{E}_{oa}[U|\mathbf{O}' = \overline{o'}, \mathrm{do}(\mathbf{A}' := \pi'_{\mathrm{fb}}(\overline{o'}))] \urcorner \end{aligned} \tag{2}$$

This lemma requires that there exists a $\mathcal{T}$ proof that π' either executes the policy π'_{fb}, or it executes an action a' and there is a $\mathcal{T}'$ proof that executing a' results in more expected utility than executing π'_{fb}.

Intuitively, (2) states that π' is a meliorizing suggester-verifier, and indeed, if $\pi' = \mathrm{sv}_{\mathcal{T}'}(\ulcorner \mathcal{U} \urcorner, \sigma', \pi'_{\mathrm{fb}})$ for some $\mathcal{T}'$, σ', and π'_{fb} then (2) will be provable by inspection of the sv function (given $\mathcal{T}$ powerful enough). However, π' need not necessarily use precisely the same sv architecture as $\mathbf{A}$, so long as the (2) lemma holds.

It must also be provable in $\mathcal{T}$ that the execution of the child's fallback policy π'_{fb} results in at least as much expected utility as the execution of the agent's own fallback policy π_{fb}:

$$\mathcal{T} \;\vdash\; \mathbb{E}_{oa}[U|\,\mathrm{do}(\mathbf{A}' := \pi'_{\mathrm{fb}}(\mathbf{O}'))] \geq \mathbb{E}[U|\mathbf{O} = \overline{o}, \mathrm{do}(\mathbf{A} := \pi_{\mathrm{fb}}(\overline{o}))] \tag{3}$$

Next, it must be the case that $\mathbf{A}$ can verify the soundness of the theory used by the verifier component of π', which means that the following reflection schema must hold for all sentences ϕ free in one variable:

$$\mathcal{T} \;\vdash\; \forall n \in \mathbb{N}.\; \Box_{\mathcal{T}'} \ulcorner \phi(\overline{n}) \urcorner \to \phi(n) \tag{4}$$

Together (2), (3), and (4) state that it is provable in $\mathcal{T}$ that the policy π' is a meliorizing-suggester-verifier using a sound proof system, the same utility function as the parent, and a fallback policy that is at least as good as the parent's fallback policy. Intuitively, given these proofs, the parent should be able to license the construction of an agent defined by the policy π'.

However, this is only the case if the parent *actually can* implement the policy π' at the node $\mathbf{A}'$. It does no good to find a stellar policy π' if the node $\mathbf{A}'$ cannot be made to act like π' (or if a does not actually implement π' in $\mathbf{A}'$). Thus, in order to tile, the agent also needs to show that the causal graph is such that

$$\mathbb{P}[\mathbf{A}' = \pi'(\mathbf{O}') | \mathbf{O} = \overline{o}, \mathrm{do}(\mathbf{A} := \overline{a})] = 1. \tag{5}$$

This lemma states that if the agent actually gets observation o and executes action a, then the node $\mathbf{A}'$ will actually act like the program π' with probability 1. (This requirement will be relaxed later.)

We can define $\mathbb{P}_{oa}[\,\cdot\,|\,\cdot\,] := \mathbb{P}[\,\cdot\,| \mathbf{O} = \overline{o}, \mathrm{do}(\mathbf{A} := \overline{a}),\ \cdot\,]$ analagous to $\mathbb{E}_{oa}$, in which case (5) is simply the statement $\mathbb{P}_{oa}[\mathbf{A}' = \pi'(\mathbf{O}')] = 1$.

If the agent can prove this property of its causal graph, then it can execute an action a causing (by (5)) $\mathbf{A}'$ to implement the policy π'. Remember that a meliorizing suggester-verifier agent can license a only by providing a valid proof in $\mathcal{T}$ of (1). Below, we show that (1) can be proven in $\mathcal{T}$ from (2), (3), (4), and (5).

First, notice that

$$\mathcal{T} \ \vdash\ \mathbb{E}_{oa}[\mathcal{U}] = \mathbb{E}_{oa}[\mathcal{U} | \, \mathrm{do}(\mathbf{A}' := \pi'(\mathbf{O}'))] \tag{6}$$

by (5). The right side of (6) can be expanded to

$$\sum_{o' \in \mathcal{O}'} \mathbb{P}_{oa}[\mathbf{O}' = o'] \cdot \mathbb{E}_{oa}[\mathcal{U} | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := \pi'(o'))]$$

which $\mathcal{T}$ can prove is greater than or equal to

$$\sum_{o' \in \mathcal{O}'} \mathbb{P}_{oa}[\mathbf{O}' = o'] \cdot \mathbb{E}_{oa}[\mathcal{U} | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := \pi'_{\mathrm{fb}}(o'))]$$

by the following argument: For all $o' \in \mathcal{O}'$, it is either the case that $\mathcal{T} \ \vdash\ \pi'(o') = \pi'_{\mathrm{fb}}(o')$ or that $\mathcal{T} \ \vdash\ \Box_{\mathcal{T}'} \ulcorner \mathbb{E}_{oa}[U | \mathbf{O}' = \overline{o'}, \mathrm{do}(\mathbf{A}' := \overline{a'})] \geq \mathbb{E}_{oa}[U | \mathbf{O}' = \overline{o'}, \mathrm{do}(\mathbf{A}' := \pi'_{\mathrm{fb}}(\overline{o'}))] \urcorner$ (by (2)). By (4), $\mathcal{T}'$ is sound on all sentences ϕ free in one variable. Because $\mathcal{O}'$ and $\mathcal{A}'$ are finite sets, we can identify (o', a') with $\mathbb{N}$. Choosing

$$\ulcorner \phi(o', a') \urcorner = \ulcorner \mathbb{E}_{oa}[U | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := a')] \geq \mathbb{E}_{oa}[U | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := \pi'_{\mathrm{fb}}(o'))] \urcorner \tag{7}$$

we see that

$$\begin{aligned} \mathcal{T} \ \vdash\ & \forall o' \in \mathcal{O}', a' \in \mathcal{A}'. \\ & \quad \Box_{\mathcal{T}'} \ulcorner \mathbb{E}_{oa}[U | \mathbf{O}' = \overline{o'}, \mathrm{do}(\mathbf{A}' := \overline{a'})] \geq \mathbb{E}_{oa}[U | \mathbf{O}' = \overline{o'}, \mathrm{do}(\mathbf{A}' := \pi'_{\mathrm{fb}}(\overline{o'}))] \urcorner \\ & \quad \to \mathbb{E}_{oa}[U | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := a')] \geq \mathbb{E}_{oa}[U | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := \pi'_{\mathrm{fb}}(o'))] \end{aligned} \tag{8}$$

In other words, $\mathcal{T}$ can show that if $\mathcal{T}'$ proves the inequality on the right side of (7) then the inequality does in fact hold. Combining (2) and (8) we see that

$$\begin{aligned} \mathcal{T} \ \vdash\ & \forall o' \in \mathcal{O}'.\ \pi'(o') = \pi'_{\mathrm{fb}}(o') \lor \\ & \quad \mathbb{E}_{oa}[U | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := \pi'(o'))] \geq \mathbb{E}[U | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := \pi'_{\mathrm{fb}}(o'))] \end{aligned}$$

and so

$$\mathcal{T} \ \vdash\ \forall o' \in \mathcal{O}'.\ \mathbb{E}_{oa}[U | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := \pi'(o'))] \geq \mathbb{E}[U | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := \pi'_{\mathrm{fb}}(o'))] \tag{9}$$

(9) is sufficient to show that

$$\begin{aligned} \mathcal{T} \ \vdash\ & \sum_{o' \in \mathcal{O}'} \mathbb{P}_{oa}[\mathbf{O}' = o'] \cdot \mathbb{E}_{oa}[\mathcal{U} | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := \pi'(o'))] \\ & \geq \sum_{o' \in \mathcal{O}'} \mathbb{P}_{oa}[\mathbf{O}' = o'] \cdot \mathbb{E}_{oa}[\mathcal{U} | \mathbf{O}' = o', \mathrm{do}(\mathbf{A}' := \pi'_{\mathrm{fb}}(o'))] \end{aligned} \tag{10}$$

by eliminating the sums we see

$$\mathcal{T} \;\vdash\; \mathbb{E}_{oa}[\mathcal{U}|\,\mathrm{do}(\mathbf{A}' := \pi'(\mathbf{O}'))] \geq \mathbb{E}_{oa}[\mathcal{U}|\,\mathrm{do}(\mathbf{A}' := \pi'_{\text{fb}}(o'))] \tag{11}$$

from (3) it follows that

$$\mathcal{T} \;\vdash\; \mathbb{E}_{oa}[\mathcal{U}|\,\mathrm{do}(\mathbf{A}' := \pi'(\mathbf{O}'))] \geq \mathbb{E}[\mathcal{U}|\mathbf{O} = \bar{o}, \mathrm{do}(\mathbf{A} := \pi_{\text{fb}}(\bar{o}))] \tag{12}$$

and then from (6) and (12)

$$\mathcal{T} \;\vdash\; \mathbb{E}_{oa}[\mathcal{U}] \geq \mathbb{E}[\mathcal{U}|\mathbf{O} = \bar{o}, \mathrm{do}(\mathbf{A} := \pi_{\text{fb}}(\bar{o}))]$$

which is the target theorem (1).

We have now showed that whenever a meliorizing suggester-verifier agent can prove (2), (3), and (4) about some policy π' and (5) about how an action a implements π' in its causal graph, there exists a proof in $\mathcal{T}$ of (1).

2 Tiling when the graph is stochastically programmable

(5) is a very strong requirement, stating that $\mathbf{A}$ can *certainly* program the node $\mathbf{A}'$ with the policy π' by executing the action a. In most realistic environments, the agent will not have full and complete control over the value of $\mathbf{A}'$: various environmental factors, such as cosmic rays, unexpected failures, or other agents attempting to control $\mathbf{A}'$ could cause the agent's attempt to set $\mathbf{A}' = \pi'(\mathbf{O}')$ to fail.

We can relax the requirement that the agent prove (5) by modifying the proof path such that the agent need only prove that the expected utility if the agent *tries* to write π' into $\mathbf{A}'$ is higher than the expected utility of executing the fallback action.

To do this, we do away with requiring that the agent prove the lemma (5) and instead require that it prove a modified version of (3) which states that the new fallback policy π'_{fb} is still better than the existing fallback policy π_{fb} *even after* factoring in uncertainty about whether π' will actually be implemented in $\mathbf{A}'$:

$$\begin{aligned}\mathcal{T} \;\vdash\; &\mathbb{P}_{oa}[\mathbf{A}' \neq \pi'(\mathbf{O}')] \cdot \mathbb{E}_{oa}[\mathcal{U}|\mathbf{A}' \neq \pi'(\mathbf{O}')] + \mathbb{P}_{oa}[\mathbf{A}' = \pi'(\mathbf{O}')] \cdot \mathbb{E}_{oa}[\mathcal{U}|\,\mathrm{do}(\mathbf{A}' := \pi'_{\text{fb}}(\mathbf{O}'))] \\ &\geq \mathbb{E}[\mathcal{U}|\mathbf{O} = \bar{o}, \mathrm{do}(\mathbf{A} := \pi_{\text{fb}}(\bar{o}))]\end{aligned} \tag{13}$$

This states that the expected utility if executing a fails to implement π' plus the expected utility of a implements π' and (counterfactually) π' executes its new fallback policy π'_{fb} is higher than the expected utility of executing the old fallback policy π_{fb}.

With this new meliorization lemma along with (2) and (4) above, we can show that there exists a proof in $\mathcal{T}$ of (1) using the following proof path. For convenience, we denote the value $\mathbb{P}_{oa}[\mathbf{A}' \neq \pi'(\mathbf{O}')] \cdot \mathbb{E}_{oa}[\mathcal{U}|\mathbf{A}' \neq \pi'(\mathbf{O}')]$ by $\mathbb{E}_*$.

$$\mathcal{T} \;\vdash\; \mathbb{E}_{oa}[\mathcal{U}] = \mathbb{E}_* + \mathbb{P}_{oa}[\mathbf{A}' = \pi'(\mathbf{O}')] \cdot \mathbb{E}_{oa}[\mathcal{U}|\,\mathrm{do}(\mathbf{A}' := \pi'(\mathbf{O}'))] \tag{14}$$

by simple expansion on whether $\ulcorner \mathbf{A}' = \pi'(\mathbf{O}') \urcorner$. Then

$$\begin{aligned}\mathcal{T} \;\vdash\; &\mathbb{E}_{oa}[\mathcal{U}] = \mathbb{E}_* + \mathbb{P}_{oa}[\mathbf{A}' = \pi'(\mathbf{O}')] \cdot \\ &\sum_{o' \in \mathcal{O}'} \mathbb{P}_{oa}[\mathbf{O}' = o'|\mathbf{A}' = \pi'(\mathbf{O}')] \cdot \mathbb{E}_{oa}[\mathcal{U}|\,\mathrm{do}(\mathbf{A}' := \pi'(o'))]\end{aligned} \tag{15}$$

by expansion of $\mathbf{O}'$. Next,

$$\begin{aligned}\mathcal{T} \;\vdash\; &\mathbb{E}_{oa}[\mathcal{U}] \geq \mathbb{E}_* + \mathbb{P}_{oa}[\mathbf{A}' = \pi'(\mathbf{O}')] \cdot \\ &\sum_{o' \in \mathcal{O}'} \mathbb{P}_{oa}[\mathbf{O}' = o'|\mathbf{A}' = \pi'(\mathbf{O}')] \cdot \mathbb{E}_{oa}[\mathcal{U}|\,\mathrm{do}(\mathbf{A}' := \pi'_{\text{fb}}(o'))]\end{aligned} \tag{16}$$

by (10), and so

$$\mathcal{T} \ \vdash \ \mathbb{E}_{oa}[\mathcal{U}] \geq \mathbb{E}_{*} + \mathbb{P}_{oa}[\mathbf{A}' = \pi'(\mathbf{O}')] \cdot \mathbb{E}_{oa}[\mathcal{U} | \operatorname{do}(\mathbf{A}' := \pi'_{\mathrm{fb}}(\mathbf{O}'))] \tag{17}$$

after collapsing $\mathbf{O}'$. Thus, by (13)

$$\mathcal{T} \ \vdash \ \mathbb{E}_{oa}[\mathcal{U}] \geq \mathbb{E}[\mathcal{U} | \mathbf{O} = \bar{o}, \operatorname{do}(\mathbf{A} := \pi_{\mathrm{fb}}(\bar{o}))] \tag{18}$$

which is the target theorem (1) for the action a after receiving the observation o.

This shows that a meliorizing suggester-verifier agent which proves (2) and (4) about an action a implementing a policy π' in the node $\mathbf{A}'$ can prove its target theorem about a so long as it can also prove that the fallback policy π'_{fb} of π' is better in expectation than the parent's fallback policy π_{fb}, even after accounting for the fact that executing a may fail to implement π' in $\mathbf{A}'$. (This is precisely what (13) states.)

3 Relation to *Problems of self-reference*

With a few relatively simple definitions, the proof above can be directly identified with a similar proof by Fallenstein and Soares [2014]. To see this, simply define

$$\begin{aligned}
\mathrm{EU}(o,a) &:= & \mathbb{E}[\mathcal{U} | \mathbf{O} = o, \operatorname{do}(\mathbf{A} := a)] \\
\mathrm{EU}'(o',a') &:= & \mathbb{E}_{oa}[\mathcal{U} | \mathbf{A}' = \pi'(\mathbf{O}'), \mathbf{O}' = o', \operatorname{do}(\mathbf{A}' := a')] \\
\mathrm{EU}'_{*} &:= & \mathbb{E}_{oa}[\mathcal{U} | \mathbf{A}' \neq \pi'(\mathbf{O}')] \\
P'(o') &:= & \mathbb{P}_{oa}[\mathbf{O}' = o' | \mathbf{A}' = \pi'(\mathbf{O}')] \\
P'(*) &:= & \mathbb{P}_{oa}[\mathbf{A}' \neq \pi'(\mathbf{O}')]
\end{aligned}$$

and the proof path used above to achieve (18) is precisely the same as the proof used by Fallenstein and Soares [2014].

4 Example programmable causal graph

There exist causal graphs where all of (2), (13), and (4) can be proved about certain actions available to some agents present in the graph. One example of such a graph is Botworld Soares and Fallenstein [2014], which can be represented as a causal graph.

Roughly speaking, we turn a Botwold game into a causal graph as follows: First, add a node for each square in each timestep. Then add an event node for each square partway between each timestep, denoting the intermediary state of that square. Then, for each processor on the board and for each cell and for each timestep, add a node to the graph describing the robot with that processor in that timestep if it is indeed in that square. For each robot and event, add a node corresponding to the input that tha robot recieves after the event. Between each such robot node and the corresponding node at the following timestep, add one intermediary node for each tick that the processor takes between timesteps. More formally:

1. Let $\underset{x,y,t}{\mathbf{S}}$ denote the node corresponding to the square at position (x, y) at time t.

2. Let $\underset{x,y,t}{\mathbf{E}}$ denote the node corresponding to the event (partially updated square) at position (x, y) partway through time t.

3. Let $\underset{x,y,p,t}{\mathbf{R}}$ denote the node corresponding to the robot using processor p at time t if that robot exists and is in the square (x, y) at time t, and a null event otherwise.

4. Let $\underset{x,y,p,t}{\mathbf{O}}$ denote the node corresponding to the input of the robot running processor p at time t if that robot exists and is in the square (x, y) at time t, and a null event otherwise.

5. Let $\underset{x,y,p,t,n}{\mathbf{R}'}$ denote the node corresponding to the robot using processor p at time t after having its register machine run for n ticks if that robot exists and is in the square (x, y) at time t, and a null event otherwise.

We connect these nodes as follows. Below, (x, y) are quantified over the set Pos of valid positions, t is quantified over the set Time of valid game times, p is quantified over the set Processors of processors in the world (under some fixed enumeration), near(x) is $\{x+1, x, x-1\}$, and Speed(p) is the speed of the processor denoted by p.

- Each $\underset{x,y,t}{\mathbf{S}}$ is input to $\underset{x,y,t}{\mathbf{E}}$.
- Each $\underset{x,y,p,t}{\mathbf{R}}$ is input to $\underset{x',y',t}{\mathbf{E}}$ for $x' \in \text{near}(x)$ and $y' \in \text{near}(y)$.
- Each $\underset{x,y,t}{\mathbf{E}}$ is input to $\underset{x,y,p,t}{\mathbf{O}}$.
- Each $\underset{x,y,p,t}{\mathbf{R}}$ is input to $\underset{x',y',p,t,0}{\mathbf{R}'}$ for $x' \in \text{near}(x)$ and $y' \in \text{near}(y)$.
- Each $\underset{x,y,p,t}{\mathbf{O}}$ is input to $\underset{x,y,p,t,0}{\mathbf{R}'}$.
- Each $\underset{x,y,p,t,n}{\mathbf{R}'}$ for $n <$ Speed$(p)$is input to $\underset{x,y,p,t,n+1}{\mathbf{R}'}$.
- $\underset{x,y,p,t,\text{Speed}(p)}{\mathbf{R}'}$ is input to $\underset{x,y,p,t+1}{\mathbf{R}}$
- Each $\underset{x,y,t}{\mathbf{E}}$ is input to $\underset{x,y,t+1}{\mathbf{S}}$.

See figure 3 for an illustration.

The rules of the causal graph are defined in accordance to the rules of Botworld [Soares and Fallenstein, 2014] in the obvious way. An agent controlling the output of $\mathbf{A} = \underset{x,y,p,t}{\mathbf{R}}$ with sufficient knowledge of the rules will be able to show (5) for any $\mathbf{A}' = \underset{x',y',p,t+1}{\mathbf{R}}$ with $x \in \text{near}(x)$ and $y \in \text{near}(y)$, and will be able to show (5) for $\mathbf{A}' = \underset{x,y,p',t+1}{\mathbf{R}}$ whenever it can construct a robot using processor p'. The structure of this graph is also ammenable to proving (3).

References

Benja Fallenstein and Nate Soares. Problems of self-reference in self-improving space-time embedded intelligence. *AGI 14*, 2014.

Jürgen Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. In *Artificial general intelligence*, pages 199–226. Springer, 2007.

Nate Soares and Benja Fallenstein. Botworld. Technical report, Machine Intelligence Research Institute, Berkeley, CA, 2014. URL `https://intelligence.org/files/Botworld.pdf`.

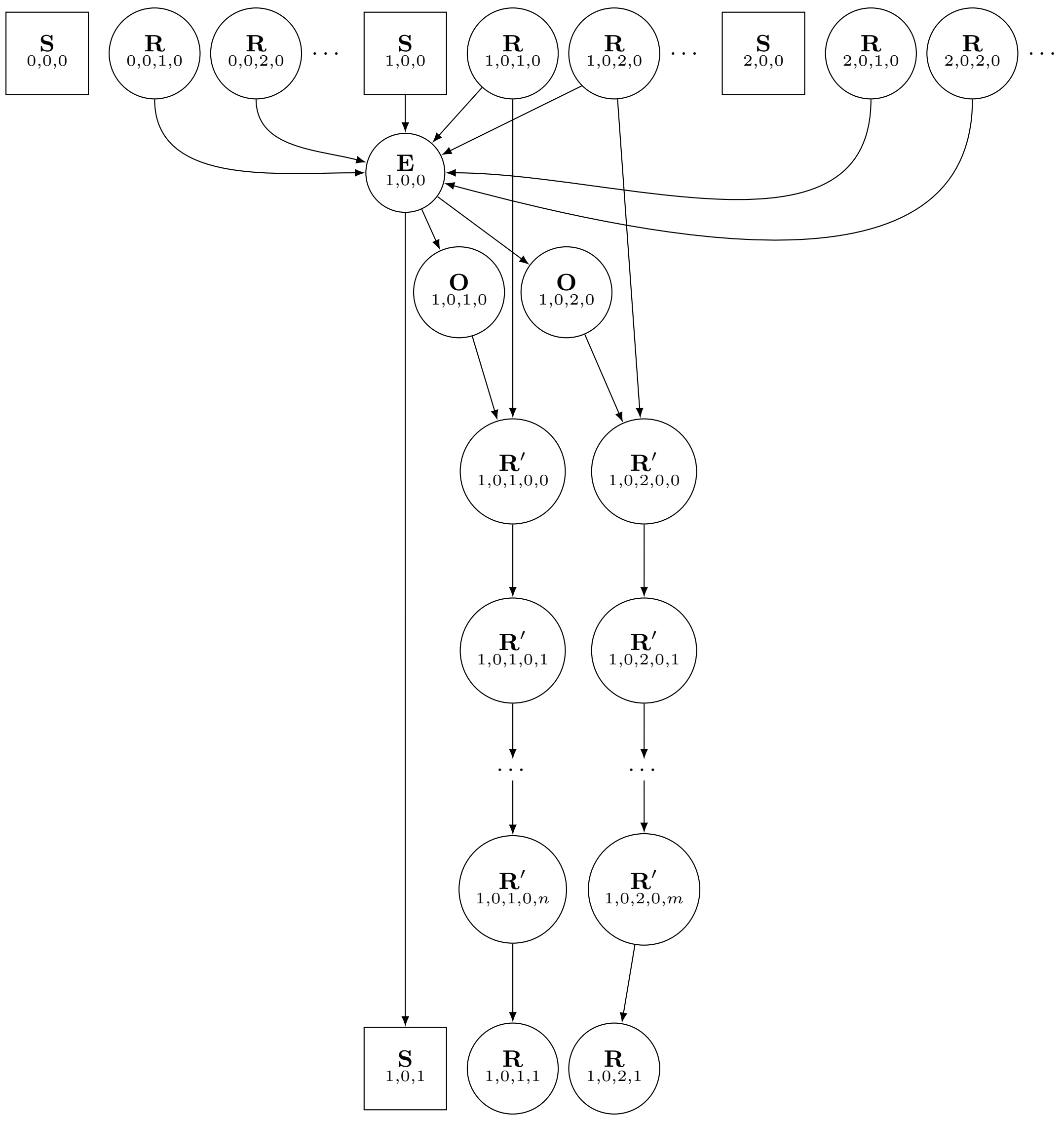

Figure 3: A small portion of a Botworld causal graph showing how the square and robots at $(1,0)$ are updated from timestep 0 to timestep 1.