# Modularity in Reinforcement Learning:
## An Algorithmic Causality Perspective on Credit Assignment

**Michael Chang**[*,†], **Sidhant Kaushik**[*,†], **Sergey Levine**[†], **Thomas L. Griffiths**[‡]

## Abstract

We present a formal definition of modularity in RL as a dynamic measure of algorithmic independence induced via credit assignment. Theoretically we prove the conditions on the agent architecture and credit assignment under which this modularity can be achieved. In particular we show that single-step temporal difference algorithms are modular for the tabular and linear function approximation settings, and that the cloned Vickrey society algorithm from the recently proposed societal decision-making framework additionally extends this modularity property to the non-linear function approximation setting. Empirically we show that RL algorithms that satisfy these conditions of modularity more efficiently transfer to new tasks whose optimal solution re-uses previously learned decision sequences.

## 1 Introduction

Gusteau's[1] taqueria has a fantastic team for making its world-class burritos: first Collete heats the tortillas, then Remy adds the meat and vegetables, and last Alfredo wraps the burrito in aluminum foil. But this week many customers fell sick because the meat was contaminated at the meat supplier's facilities. The taqueria received many angry reviews – an undesirable outcome for everyone.

How could the taqueria recover from this mishap? Clearly, Remy should now stop adding meat to the burritos, perhaps replacing them with tofu. But should Collete also stop heating tortillas, or Alfredo stop wrapping aluminum foil? The angry reviews, after all, were undesirable for all of them: should the credit assignment of the angry reviews affect all of them equally? Intuitively, no. Given a burrito order, the *decisions* to heat tortillas and wrap aluminum foil are independent of the *decision* to stuff burritos with meat. Moreover, the customer dissatisfaction in tacos should have no reflection on the quality of the taqueria's cheese quesadillas, for which Collete's tortilla skills and Alfredo's wrapping skills are useful. To abandon tortilla heating or foil wrapping because of the burrito disaster would harm the team's abilities to make other menu items that customers enjoy.

As the above example illustrates, when a singular event causes a well-functioning system to fail, the modular solution is for credit assignment to be isolated to the scope of that event without having to expensively update the entire system. For reinforcement learning (RL), achieving such modularity in credit assignment requires maintaining the independence among decisions in different contexts when performing an update to the weights of the learner. Intuitively, if the learner need only need change a single decision within an otherwise previously learned optimal sequence of decisions, the hypothesized implication of enforcing such independence is better sample efficiency in transfer.

To study this intuition precisely, this paper (1) formalizes a definition of modularity in RL as a dynamic measure of algorithmic independence induced via credit assignment, (2) theoretically proves the conditions on the agent architecture and credit assignment mechanism under which this modularity can be achieved, and (3) empirically shows that RL algorithms that satisfy these conditions

* Equal contribution. [†]University of California Berkeley. [‡]Princeton University.

[1] Credits to [6].

Presented at Deep RL Workshop, NeurIPS 2020.

more efficiently transfer to new tasks whose optimal solution re-uses previously learned decision sequences. Our formalism draws upon the language of algorithmic causality [15] and the recently proposed societal decision-making [9] framework of RL. We present, to the best of our knowledge, the first theoretical connection between credit assignment and causality in RL, which we call the algorithmic causal model of reinforcement learning (ACRL). We use ACRL to systematically analyze the modularity of all major classes of standard RL algorithms.

Theoretically, we prove that for the major classes of RL algorithms, modularity can be satisfied if and only if the agent architecture is *structurally local* – the weights of the agent architecture factorizes along the set of possible decisions – and the credit assignment mechanism is *temporally local* – the supervision for updating a decision made at a particular state depends only on information from that state transition. We show that single-step temporal difference algorithms are modular for the tabular and linear function approximation settings, and that the cloned Vickrey society (CVS) algorithm from the societal decision-making (SDM) framework additionally extends this modularity property to the non-linear function approximation setting. Empirically, we show evidence that modular RL algorithms are more sample efficient for transfer learning in settings whose optimal solution involve re-using previously learned subsequences of optimal decisions, giving them a flexibility for recovering from the kinds of unexpected changes in the environment that motivated the beginning of this section.

## 2 Related Work

This paper synthesizes perspectives from multi-agent reformulations on intelligence [4, 5, 9, 22, 34], computer programming [20, 23, 24], and causality [14, 27, 28] around the theme of algorithmic independence [15, 17, 19] to present a formalism for modularity in RL that reframes how RL is traditionally conceptualized [36] in several ways. The first conceptual shift is to treat the *decision-mechanism* (§4.1), rather than the agent, as the core primitive of decision making, building directly off the societal decision-making framework [9] that refactors the traditional monolithic agent into a multi-agent system of mechanisms that each control a different decision. The practical benefit of a multi-agent reframing of a traditionally monolithically-framed problem enables the credit assignment [21] into individual decisions to be decoupled from that into others, which motivates the second conceptual shift: that modularity is a property of not merely how the system itself is statically factorized – the view taken by most previous work on modularity in machine learning [2, 3, 10, 11, 13, 16, 25], but how the outer process that *changes* the system is factorized as well. In computer programming this outer process is the human reasoning process [12]; in RL it is the credit assignment process. To analyze how the causal structure of the credit assignment process interacts with the causal structure of the decision-making system that it modifies requires a third conceptual shift based on the duality of "code as data" [1] that embeds in the *same* causal graph both the decision mechanisms that transform states in the Markov decision process and the credit assignment (meta-)mechanism that transforms the decision mechanisms themselves in the learning process.

## 3 Preliminaries

This section reviews a key result from algorithmic causality and builds upon the recently proposed societal decision-making framework [9] to introduce the concept of the *decision mechanism* as a unifying lens for viewing RL algorithms in subsequent sections.

### 3.1 Algorithmic causality

We paraphrase Thm. 4 from [15], which generalizes structural causal models [26, 27] to general programs, allowing us to use the language of causality to assess the independence of computable objects by inspecting the structure of the generative program that produced them. Algorithmic independence is defined in terms of algorithmic mutual information: $x \perp\!\!\!\perp y|z \Leftrightarrow I(x:y|z) \stackrel{+}{=} 0$.

**Definition 3.1 (algorithmic model of causality).** *Let $\mathcal{G}$ be a directed acyclic graph (DAG) formalizing the causal structure among the strings $x_1, ..., x_N$. Every $x_j$ is computed by a program $\mathtt{F}_j$ with length $O(1)$ from its parents $\{pa_j\}$ and additional noise input $n_j$. Assume the noise $n_j$ are jointly independent: $n_j \perp\!\!\!\perp \{n_{\neq j}\}$. Formally, $x_j := \mathtt{F}_j(\{pa_j, n_j\})$, meaning that the Turing machine computes $x_j$ from the input $\{pa_j\}, n_j$ using the additional program $\mathtt{F}_j$ and halts.*

The **algorithmic causal Markov condition** states $d$-separation implies conditional independence:

**Theorem 3.1** (**algorithmic causal Markov condition**). *Consider the DAG $\mathcal{G}$ from Def. 3.1. Let $\{nd_j\}$ be the concatenation of all non-descendants of $x_j$ except itself. Then, $x_1, ..., x_N$ satisfy $x_j \perp\!\!\!\perp \{nd_j\}|\{pa_j\}$.*

We also assume its converse, that all independencies are accounted for in the causal structure of $\mathcal{G}$:

**Postulate 3.2** (**faithfulness** [33]). *Consider the DAG $\mathcal{G}$ from Def. 3.1. Given three sets $S$, $T$, $R$ of nodes, $I(S:T|R) \overset{+}{=} 0$ implies $R$ d-separates $S$ and $T$.*

### 3.2 Societal decision-making

Sequential decision-making can be formalized with a Markov decision process (MDP), defined by state space $\Omega_S$, action space $\Omega_A$, transition function $\mathtt{T} : \Omega_S \times \Omega_A \to \Omega_S$, reward function $\mathtt{R} : \Omega_S \times \Omega_A \to \mathbb{R}$, and discount factor $\gamma$. The MDP objective is to maximize the return $\sum_{t=0}^{T} \gamma^t \mathtt{R}(s_t, a_t)$.

In the societal decision-making (SDM) framework [9], decisions are made by a society of $N$ agents $\alpha^n$. Each agent is a tuple $\alpha^n = (\mathtt{D}^n, \mathtt{O}^n)$ of a **decision mechanism** $\mathtt{D}^n : \Omega_S \to \Omega_B$ and a fixed **transformation mechanism** $\mathtt{O}^n : \Omega_S \to \Omega_S$. Given state $s$, each decision mechanism produces a bid $b_s^n := \mathtt{D}^n(s)$ in a bidding space $\Omega_B$ that indicates the applicability of the $n$th agent to operate on state $s$. The society uses a **selection mechanism** $\mathtt{M} : \Omega_B^N \to \{1, ..., N\}$ that selects an agent given the bids of all agents. The selected agent's transformation mechanism transforms $s$ into the next state $s'$. Thus instead of the transition and reward functions $\mathtt{T}$ and $\mathtt{R}$, SDM uses $\mathtt{T}^{\mathtt{O}} : \{1, ..., N\} \times \Omega_S \to \Omega_S$ and $\mathtt{R}^{\mathtt{O}} : \{1, ..., N\} \times \Omega_S \to \mathbb{R}$ to compute the next state and reward.

**Cloned Vickrey society** [9] introduced the cloned Vickrey society (CVS) RL algorithm that interprets the Bellman optimality equation as an economic transaction between agents seeking to optimize their utilites in a Vickrey auction [38] at each time-step. The auction selection mechanism selects the highest bidding agent $i$, which receives a utility

$$\underbrace{U^i_{s_t}(\alpha^{1:N})}_{\text{utility}} = \underbrace{\mathtt{R}^{\mathtt{O}}(\alpha^i, s_t) + \gamma \cdot \max_k b^k_{s_{t+1}}}_{\text{revenue, or valuation } v_{s_t}} - \underbrace{\max_{j \neq i} b^j_{s_t}}_{\text{price}}, \tag{1}$$

and the rest receive a utility of 0. In CVS each agent bids twice: the highest and second highest bids are produced by the same weights. The Vickrey auction incentivizes each agent to truthfully bid the $Q$-value of its associated transformation mechanism, independent of the identities and bidding strategies of other agents.

## 4 Defining Modularity in RL

Understanding modularity in RL requires us to formalize what we mean by "independent decisions" in a way that applies to any RL algorithm. We propose to formalize this independence as the independence of the decision mechanisms themselves. This paper restricts the scope of discussion to the major classes of standard RL algorithms in [36] – precisely action-value (of which CVS is an instance), policy gradient, and actor-critic methods – for discrete decisions. First, we show how to view these classes of algorithms from the perspective of decision mechanisms. Next, we formally define modularity in RL as a criterion on the dependencies induced by credit assignment among decision mechanisms of a decision sequence.

### 4.1 Decision mechanisms

We show how to view $Q$-functions, parameterized policies, and actor-critic architectures from the perspective of decision mechanisms. Most generally, a decision mechanism $\mathtt{D}^n$ is a program that produces a bid $b_s^n$ given state $s$. We can view action probabilities in policy gradient methods as bids by adding the constraint that $b_s^n \in [0, 1]$ and $\sum_n b_s^n = 1$ and set the selection mechanism to be a sampler. Action-value methods treat the bids as estimated $Q$-values and set the selection mechanism to depend on the exploration strategy ($\mathtt{M} = \max$ if exploration is greedy). A bid for actor-critic methods is a tuple of an action probability and a $Q$-value, and the selection mechanism samples over the action probabilities across all the bid tuples. These methods are often viewed as making decisions

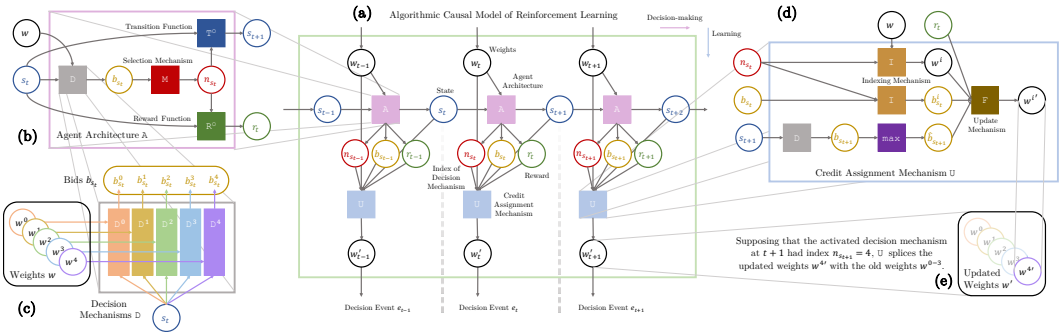


Figure 1: **The algorithmic causal model of reinforcement learning for modular RL algorithms.** **(a)** The computation graph $\mathcal{G}^{\text{RL}}$ of a modular RL algorithm is a trellis over states and weights. **(b)** The agent architecture $\texttt{A}$. **(c)** The weights $w^i$ for the decision mechanisms $\texttt{D}^i$ are decoupled. **(d)** The inputs to the credit assignment mechanism $\texttt{U}$ are isolated to only the information at a single decision event. What is shown is the Bellman update for updating $b_{s_t}$ as the estimated $Q$-value for state $s_t$ and decision mechanism $i$. **(e)** The updated weights $w^{i\prime}$ of the active decision mechanism at $e_t$ are spliced back with the old weights of the other decision mechanisms.

over literal actions, but [37] abstracted such decision-making to be over any transformation of state (i.e. options) of which literal actions are only a special case. In anticipation of the causal language we use later, henceforth we adopt the terminology of the decision mechanism $\texttt{D}^n$ and the transformation mechanism $\texttt{O}^n$ to generalize over any RL algorithm for making decisions at any level of abstraction.

### 4.2 Decision events and decision sequences

Each decision mechanism $\texttt{D}^n$ is encoded by weights $w^n$. A **decision event** $e := \{s, w_s, n_s, b_s, s', r\}$ is a set containing all information about a decision at a single time-step, not unlike what is stored in replay buffers: the state $s$, the weights of all decision mechanisms $w_s$, the bids of the decision mechanisms $b_s = [b_s^1, ..., b_s^N]$, the index $n_s$ of the decision mechanism whose associated transformation mechanism $\texttt{O}^n$ was selected by the selection mechanism $\texttt{M}$, the next state $s'$ produced $\texttt{O}^n$ applied to $s$, and the reward $r$ received from the environment for this state transformation. Define a **decision sequence** $\bar{e} := (e_1, ..., e_T)$ as any consecutive sequence of decision events $e_t = (s_t, w_{s_t}, n_{s_t}, b_{s_t}, s'_t, r_t)$ within a continuous interaction with the environment, agnostic to whether the interaction is in an episodic or non-episodic setting. The indices $n_{s_{1:T}}$ denote the **active** decision mechanisms in $\bar{e}$.

### 4.3 Credit assignment, independence, and modularity

We formalize modularity in RL in terms of the dependencies induced by the credit assignment process among the decision mechanisms of a decision sequence. Given a decision sequence $\bar{e}$, note that we assume that the weights $w_{s_t}$ used in all decision events $e_t \in \bar{e}$ are the same; thus we use $w_s$ and $w$ interchangeably. We define a **credit assignment update** $\bar{\texttt{U}}$ as an operation that produces a new set of weights $w'$ from $w$ given $\bar{e}$. Intuitively, if $\bar{\texttt{U}}$ produces the new weights $w^{i\prime}$ and $w^{j\prime}$ of two decision mechanisms $\texttt{D}^i$ and $\texttt{D}^j$ using the same information (e.g. credit assignment using Monte Carlo returns updates all decision mechanisms selected in $\bar{e}$ with the same information $\sum_{t=1}^{T} r_t$), we would expect $\bar{\texttt{U}}$ to induce some dependence between $\texttt{D}^i$ and $\texttt{D}^j$. Making this intuition precise, we define modularity as the conditional algorithmic mutual information among the updated decision mechanisms:

**Definition 4.1** (**modularity in reinforcement learning**). *Let $w^{i\prime}_{s_t}$ be the new weights produced by $\bar{\texttt{U}}$ of the decision mechanism $\texttt{D}^i$ that was active at state $s_t$ of $\bar{e}$. An RL algorithm is **modular** if for all $w^{j\prime}_{s_{\tilde{t}}}$ where $\tilde{t} \neq t$ and $j \neq i$,*

$$I\left(w^{i\prime}_{s_t} : w^{j\prime}_{s_{\tilde{t}}} \middle| s_t, s_{t+1}, w\right) \stackrel{+}{=} 0. \tag{2}$$

Def. 4.1 precisely captures the intuition that modularity is a dynamic measure of independence, or lack of mutual information, of the components of a system when they undergo change. In the context of RL, this change is produced via credit assignment, a computational process external to the decision mechanisms that modifies their weights. By viewing a decision sequence as an execution trace of a decision-making program, where each decision event is the subtrace of a subprogram that computes

the consequence of a single decision, Def. 4.1 states the criterion for the credit assignment process to "program" these subprograms in a modular way: with the past weights $w$ as background information, changing the functionality of a particular subprogram should not affect the other subprograms conditioned on the interface between them, given by the input $s_t$ and output $s_{t+1}$. This invariance of certain components to changes in other components is what make software systems evolvable and extensible. We next develop a toolset based on algorithmic causality for assessing whether an RL algorithm meets these desirable properties of modularity.

## 5 An Algorithmic Causal Model of RL

This section takes seriously the analogy of relating modularity in RL algorithms to modularity in programs alluded to in the previous section and presents the **algorithmic causal model of reinforcement learning** (ACRL), a model of RL for assessing the modularity of RL algorithms directly from the algorithmic causal structure of the RL algorithm itself.

The key idea is simple: express both the decision-making process and credit assignment process as programs under the same causal graph. The algorithmic graphical structure of any RL algorithm is an unrolled double trellis $\mathcal{G}^{\text{RL}}$ that defines a partial ordering over states and weights. Decision-making generates a trellis over states that constitutes a decision sequence $\bar{e}$. Define the **agent architecture** $\texttt{A}$, encoded by weights $w$ and whose input is $s$ and output is $(s', r)$, as the operation that generates each step in this trellis of states. Credit assignment operates on the encoding of $\texttt{A}$ itself and generates a trellis over weights that constitutes $\texttt{A}$'s learning trajectory. $\bar{\texttt{U}}$ is the operation that generates each step in this trellis of weights and let $\texttt{U}$ denote the **credit assignment mechanism** that modifies the weight from $w_t^i$ to $w_t^{i\prime}$ of a particular decision mechanism $\texttt{D}^i$ activated at a particular time-step $t$ in the decision sequence. $\texttt{A}$ is a mechanism that transforms states, and $\texttt{U}$ (and by extension $\bar{\texttt{U}}$) is a *meta-mechanism* [18] that transforms $\texttt{A}$: the weights $w$ that encode $\texttt{A}$ constitute immutable background knowledge at the level of decision-making but mutable variables at the level of credit assignment.

**Proposition 5.1 (algorithmic causal model of RL).** *Let $\mathcal{G}^{RL}$ specify the complete algorithmic graphical structure of any reinforcement learning process. The nodes of $\mathcal{G}^{RL}$ are the variables $\{s, w_s, n_s, b_s, s', r\}$ of all decision events during learning. These nodes are computed via algorithmically independent programs with length $O(1)$ that constitute the subprograms that implement $\texttt{A}$ and $\texttt{U}$. Then the nodes satisfy the algorithmic causal Markov condition under $\mathcal{G}^{RL}$.*

The significance of Prop. 5.1 is that it establishes the link between algorithmic causality and RL: comparing the modularity of different RL algorithms reduces to directly inspecting the $d$-separation properties of $\mathcal{G}^{\text{RL}}$, whose causal structure depends on how the RL algorithm defines the causal structure of $\texttt{A}$ and $\texttt{U}$. The algorithmic model of causality is needed because the credit assignment mechanism is typically formulated as a deterministic operation, for which the standard probabilistic model of causality is not defined. ACRL thus imports the entire causal toolkit to analyze any computable RL algorithm. Whereas from the perspective of $\texttt{A}$, a credit assignment update constitutes an exogenous intervention on the decision mechanisms, externalizing the encodings $w$ of the decision mechanisms as endogenous variables in the same causal graph enables us to perform causal analysis on the credit assignment itself.

## 6 The Modularity of RL Algorithms

This section uses ACRL to systematically assess the modularity of all major classes of RL algorithms. This requires us to determine the causal structure of the agent architecture $\texttt{A}$ and of the credit assignment mechanism $\texttt{U}$. We first classify all RL algorithms on whether their agent architecture is **structurally local** – i.e. the decision mechanisms do not share weights. We next classify all RL algorithms on whether their credit assignment mechanism is **temporally local** – i.e. the supervision to update the active decision mechanism for decision event $e$ can be computed solely from the information in $e$. We then claim that RL algorithms that are both structurally and temporally local are modular, and highlight $Q$-learning and the cloned Vickrey society as notable examples of modular RL algorithms in the tabular and function approximation settings, respectively. We do not assume any other supervision signal to the weights, such as regularization on their complexity, other than given by the standard definitions of the RL algorithms we consider. Proofs are in the appendix.

### 6.1 Structural Locality

Before considering credit assignment, we first consider how the agent architecture itself induces a dependence between the decision mechanisms.

**Definition 6.1** (**structural locality**). *An agent architecture is **structurally local** if the decision mechanisms do not share weights with non-negligible complexity.*

We assume in this paper that the weights of any decision mechanism have non-negligible complexity from their random initialization, and that they maintain non-negligible complexity throughout learning. The importance of structural locality is that the lack of this property enables us to rule out the independence of decision mechanisms at any point during the learning process:

**Proposition 6.1** (**independence $\Rightarrow$ structural locality**). *If an agent architecture is not **structurally local**, then its decision mechanisms are never mutually independent conditioned on its causal ancestors in $\mathcal{G}^{RL}$.*

A clear example of such weight sharing is in the deep function approximators used for parameterizing deep policies or deep $Q$ functions. However, even the decision mechanisms of parameterized policies *without* function approximation are unconditionally dependent because they all share the same normalizing factor, whereas the decision mechanisms of $Q$-functions with *linear* function approximation are *not* unconditionally dependent because each decision mechanism is parameterized by an independent linear map.

### 6.2 Temporal locality

Temporal locality concerns the source of information used by the credit assignment mechanism to update the active decision mechanism $\texttt{D}^i$ for a particular decision event.

**Definition 6.2** (**temporal locality**). *Let $z_t$ be the input into the credit assignment mechanism $\texttt{U}$ for updating the active decision mechanism at decision event $e_t$. $\texttt{U}$ is **temporally local** if $z_t \subseteq e_t$.*

In particular, temporal locality implies that a subset of $s_t, w_{s_t}, n_{s_t}, b_{s_t}, s_{t+1}, r_t$ are the only causal parents of $w^{i'}_{s_t}$, the updated weights of $\texttt{D}^i$. Under this definition, policy gradient and action-value methods that use Monte Carlo returns, which use rewards from multiple time-steps of the decision sequence rather than just the from $e_t$, for estimating the $Q$-function, are not temporally local. Nor are methods that use TD-$\lambda$ [35] or generalized advantage estimation [29, GAE], which use both rewards and bids from multiple time-steps, temporally local either. However, vanilla actor-critic methods are temporally local, as the critic is trained with a Bellman update that can be computed from $s_t, r_t, s_{t+1}$, and the policy is trained with the critic's output, which is part of the $b_t$. With the same reasoning, all single-step temporal difference (TD-0) algorithms like $Q$-learning and CVS are also temporally local. To our knowledge, all temporally local RL algorithms use the Bellman update that involves computing a target bid $\hat{b}_{s_{t+1}}$ from $s_{t+1}$. For $Q$-learning and CVS, $\hat{b}_{s_{t+1}} = \max_k b^k_{s_{t+1}}$.

### 6.3 Modularity

Structural locality is a property of the causal structure of the agent architecture. Temporal locality is a property of the causal structure of the credit assignment mechanism. Taken together, we have:

**Theorem 6.2** (**modularity**). *An RL algorithm is modular if and only if its agent architecture is structurally local and credit assignment mechanism is temporally local.*

For modular RL algorithms, $e_t$ $d$-separates the updated weights $w^{i'}_{s_t}$ of the active decision mechanism $\texttt{D}^i$ from the updated weights $w^{j'}_{s_{\tilde{t}}}$ of all other decision mechanisms $\texttt{D}^j$ at other time-steps $\tilde{t}$. In the tabular setting, all TD-0 algorithms are modular because each cell in the $Q$-table is independently updated. In the function approximation setting, modularity can be achieved with linear function approximators or by factorizing the $Q$-function along the decision mechanisms, as CVS does.

At this point we have established our main result. Def. 4.1 proposed that modularity is a property of the *process* itself by which the credit assignment mechanism $\texttt{U}$ iteratively programs the source code of the agent $\texttt{A}$ (i.e. by modifying the weights of the decision mechanisms). Thm. 6.2 stated the conditions on the causal structure of the agent architecture and the credit assignment mechanism for this property

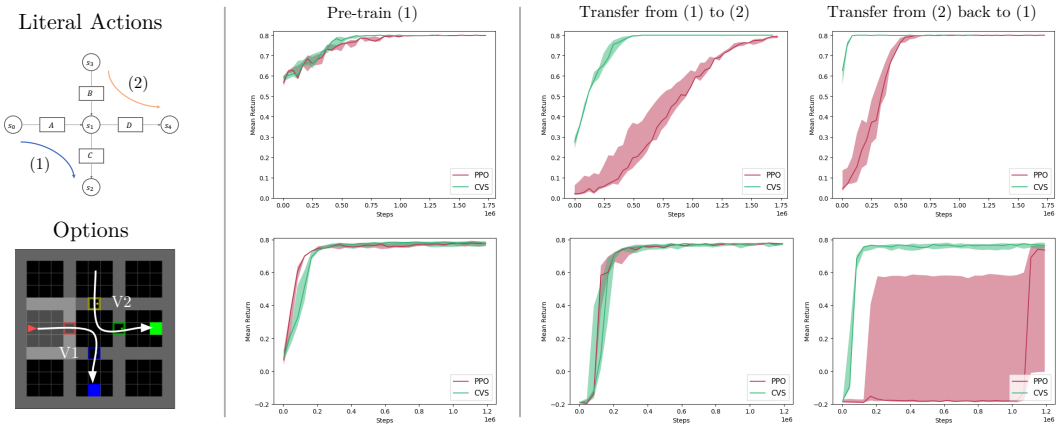


Figure 2: **Task interference.** We are interested in the extent that modularity preserves the optimality of the optimal decision mechanisms for task 1 even after it trains to convergence on task 2. When transferring back to task 1 from task 2 (right column), the modular method (CVS) is significantly more sample efficient in both the tabular (top) and Minigrid (bottom) instantiations of the transfer task.

to hold true in general. Though algorithmic mutual information is generally incomputable, a main benefit of ACRL is that it provides a practical way to use the structure of the computation graph of the RL algorithm itself to encode algorithmic independence, which holds at the level of the atomic operations of the programming language the algorithm is implemented in.

# 7 Simulations

We now return back to the kinds of transfer settings that motivated this paper: those that involve re-using optimal decisions learned previously for new tasks. We empirically show a significant improvement in transfer performance of modular over non-modular RL algorithms under two transfer settings. The first (§7.1) tests robustness to task interference. The second (§7.2) tests recovery under a sparse perturbation on the transition function $\mathtt{T}^0$ of the MDP (i.e. a change in $\mathtt{T}^0$ at a single state).

We compare two on-policy algorithms: PPO [30] and CVS [9]. PPO is an actor-critic algorithm that learns a parameterized policy with GAE; thus PPO is neither structurally nor temporally local. CVS is a modular action-value method. We consider transformation mechanisms both as literal actions and as options. Whereas our theoretical result concerns the mutual independence of updated decision mechanisms within only a single credit assignment update, our experiments test to what extent this independence maintains *across* credit assignment updates the optimality of decisions that do not need to be changed during transfer.

## 7.1 Task interference

We consider the following setup for studying robustness to task interference: first the learner trains on task 1, then transfers to task 2, then transfers back to task 1 again. The optimal decision mechanisms of task 1 are disjoint from the optimal decision mechanisms of task 2. We are interested in the extent that modularity preserves the optimality of the optimal decision mechanisms for task 1 even after it trains to convergence on task 2 – this is an important problem that continual learning systems face. We measure this extent of preservation by the sample efficiency in transferring back from task 2 to task 1. Indeed, as shown in Fig. 2, CVS converges with more than three times fewer samples than PPO when transferring from task 1 to task 2, and converges with about six times fewer samples than PPO when transferring back from task 2 to task 1.

## 7.2 Recovery from isolated changes in the environment

Now we consider transfer scenarios where the optimal performance on the transfer task requires making an isolated change at a single decision event of a decision sequence while keeping the rest of the behavior unchanged. This is exactly the kind of problem that our friends at Gusteau's taqueria

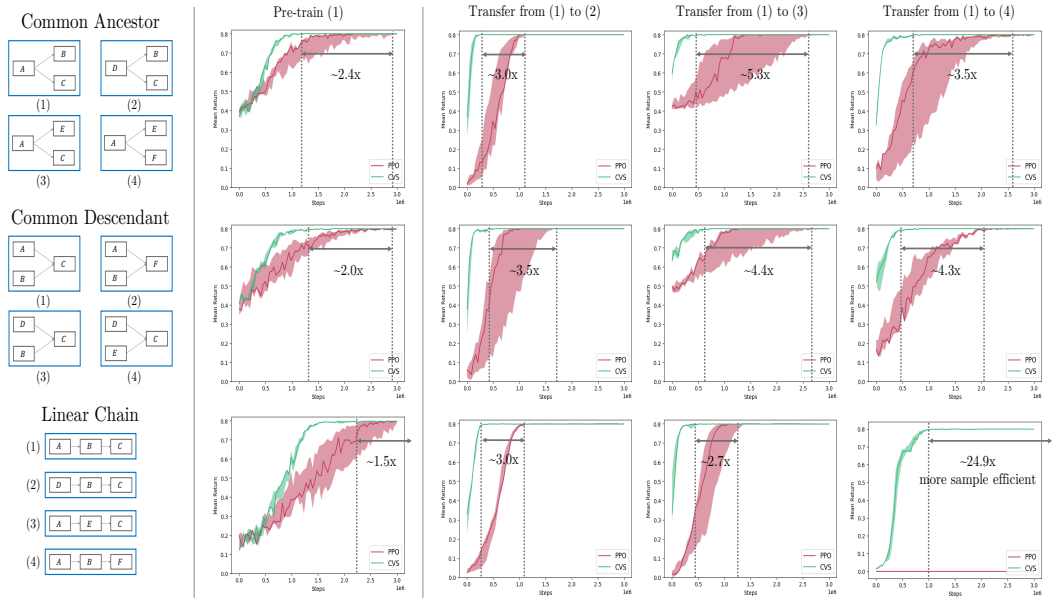


Figure 3: **Recovery from isolated changes in the environment.** For each task topology (leftmost column) we have a pre-training task, labeled (1) and three independent transfer tasks, labeled (2-4). Each transfer task represents a different way to modify the MDP used for pre-training. CVS consistently exhibits higher sample efficiency in transfer over PPO, at one point reaching 24.9x more sample efficient (bottom right plot). We set the convergence time as the first time after which the return deviates by no more than $\varepsilon = 0.01$ from the optimal return, 0.8, for 30 epochs of training. Each epoch is a set of credit assignment updates on a new batch of 4096 decision events recorded through environment interaction.

(§1) faced and an important problem to solve to build intelligent agents that can flexibly re-use past knowledge for solving new problems. More precisely, the MDP in the transfer task is derived from the MDP in the pre-training task by intervening on either the starting state distribution $\mathtt{S}$, transition function $\mathtt{T}^{\mathtt{O}}$, or the reward function $\mathtt{R}^{\mathtt{O}}$ such that the optimal decision mechanism in one state has changed, while the rest of the MDP remains the same.

In the same way that $d$-separation is conducted with triplets of nodes, we exhaustively enumerated all possible topologies of triplets of decisions: linear chain, common ancestor, and common descendant (Fig. 3). For each topology we exhaustively enumerated all ways of making an isolated change to an optimal decision sequence. Note that the common ancestor and common descendant topologies involve multi-task training for two decision sequences of length two, while linear chain involves single-task training for a decision sequence of length three. Thus for each topology we have a pre-training task and three independent transfer tasks that each start with the learner from the pre-training task, where each transfer task represents a different way to modify the MDP used for pre-training.

We then test the extent that modularity improves the recovery from such isolated changes by measuring the sample efficiency in each transfer task. Across all experiments, the trend we observe is the same: the modular CVS significantly outperforms PPO in transfer sample efficiency. This increased sample efficiency in transfer (in one instance 20x, bottom right of Fig. 3) is most often greater than the increase in sample efficiency during pre-training (2x). One explanation for the worse sample efficiency of PPO is that the dependence among its weights during credit assignment causes it to unnecessarily unlearn decisions that should still be optimal when it receives low reward initially during transfer.

## 8 Discussion

We have formalized modularity in RL as a criterion for the algorithmic independence of decision mechanisms during credit assignment, assessed several major classes of RL algorithms along this criterion, and shown empirical evidence that modularity improves a learner's ability to flexibly adapt to new tasks by re-using optimal decisions learned previously. We hope that this provides a foundation for future work studying credit assignment from the perspective of algorithmic causality.

## Broader Impact

This work applies the causal toolkit to analyze the algorithmic independencies induced by credit assignment. To the best of our knowledge it is the first to analyze the computational graph of a machine learning algorithm as a causal graph. This type of analysis could potentially open paths for new research that study the dependencies in the parameters of a learner induced by a supervision signal through credit assignment. Understanding the causal structure of learning agents, as well as their learning processes, can potentially further developments in improving their generalization and interpretability.


## Acknowledgments and Disclosure of Funding

We thank Michael Janner, Aviral Kumar, and Jason Peng for insightful discussions. This research was supported in part by the DARPA L2M program and AFOSR grant FA9550-18-1-0077. MC is supported by the National Science Foundation Graduate Research Fellowship. SK was supported in part by Berkeley Engineering Student Services.


## References


[1] H. Abelson and G. J. Sussman. *Structure and interpretation of computer programs*. The MIT Press, 1996.

[2] F. Alet, T. Lozano-Pérez, and L. P. Kaelbling. Modular meta-learning. *arXiv preprint arXiv:1806.10166*, 2018.

[3] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48, 2016.

[4] D. Balduzzi. Cortical prediction markets. *arXiv preprint arXiv:1401.1465*, 2014.

[5] E. B. Baum. Toward a model of mind as a laissez-faire economy of idiots. In *ICML*, pages 28–36, 1996.

[6] B. Bird, B. Lewis, J. Pinkava, and J. Capobianco. Ratatouille, 2007.

[7] G. J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM (JACM)*, 13(4):547–569, 1966.

[8] G. J. Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM (JACM)*, 22(3):329–340, 1975.

[9] M. Chang, S. Kaushik, S. M. Weinberg, T. L. Griffiths, and S. Levine. Decentralized reinforcement learning: Global decision-making via local economic transactions. *arXiv preprint arXiv:2007.02382*, 2020.

[10] M. B. Chang, A. Gupta, S. Levine, and T. L. Griffiths. Automatically composing representation transformations as a means for generalization. *arXiv preprint arXiv:1807.04640*, 2018.

[11] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2169–2176. IEEE, 2017.

[12] E. W. Dijkstra. Letters to the editor: go to statement considered harmful. *Communications of the ACM*, 11(3):147–148, 1968.

[13] A. Goyal, A. Lamb, J. Hoffmann, S. Sodhani, S. Levine, Y. Bengio, and B. Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

[14] D. M. Hausman and J. Woodward. Independence, invariance and the causal markov condition. *The British journal for the philosophy of science*, 50(4):521–583, 1999.

[15] D. Janzing and B. Schölkopf. Causal inference using the algorithmic markov condition. *IEEE Transactions on Information Theory*, 56(10):5168–5194, 2010.

[16] L. Kirsch, J. Kunze, and D. Barber. Modular networks: Learning to decompose neural computation. In *Advances in Neural Information Processing Systems*, pages 2414–2423, 2018.

[17] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of information transmission*, 1(1):1–7, 1965.

[18] J. Lemeire and D. Janzing. Replacing causal faithfulness with algorithmic independence of conditionals. *Minds and Machines*, 23(2):227–249, 2013.

[19] M. Li, P. Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008.

[20] B. H. Liskov. A design methodology for reliable software systems. In *Proceedings of the December 5-7, 1972, fall joint computer conference, part I*, pages 191–199, 1972.

[21] M. Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[22] M. Minsky. *Society of mind*. Simon and Schuster, 1988.

[23] D. L. Parnas. Information distribution aspects of design methodology. 1971.

[24] D. L. Parnas. On the criteria to be used in decomposing systems into modules. In *Pioneers and Their Contributions to Software Engineering*, pages 479–498. Springer, 1972.

[25] D. Pathak, C. Lu, T. Darrell, P. Isola, and A. A. Efros. Learning to control self-assembling morphologies: a study of generalization via modularity. In *Advances in Neural Information Processing Systems*, pages 2295–2305, 2019.

[26] J. Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

[27] J. Pearl. *Causality*. Cambridge university press, 2009.

[28] J. Peters, D. Janzing, and B. Schölkopf. *Elements of causal inference*. The MIT Press, 2017.

[29] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[31] R. J. Solomonoff. A preliminary report on a general theory of inductive inference. United States Air Force, Office of Scientific Research, 1960.

[32] R. J. Solomonoff. A formal theory of inductive inference. part ii. *Information and control*, 7(2):224–254, 1964.

[33] P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[34] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber. Compete to compute. In *Advances in neural information processing systems*, pages 2310–2318, 2013.

[35] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[36] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2020.

[37] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[38] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

# A Background

We review key results in algorithmic causality to establish a starting point for our theoretical results.

## A.1 Notation and terminology

We assume a universal Turing machine that executes programs, denoted with capital monospace (P), whose inputs and outputs are binary strings $x$, denoted with lowercase script. We assume programs are stateless. We use $=$ to denote equality and $:=$ to denote assignment. We use $\stackrel{+}{=}$ to denote equality up to an additive constant (and similarly for inequalities), where the additive constant comes not from the properties of the quantities on both sides of the equation but on the properties of the Turing machine.

We denote a random variable by a capital letter (e.g. $X$) with event space $\Omega_X$ and distribution $\mathbb{P}(X)$. Assuming computable probability distributions, we overload notation to use $x$ to represent a string, if the program is deterministic, or a sample, if the program is stochastic. Thus samples of $X$ are denoted by $x$ and the probability that $X = x$ by $P_X(x)$. We can view probabilistic and algorithmic causal models under the same language of algorithmic causality by noting that sample $x$ of random variable $X$ is a string produced by a program composed of a mechanism that computes $\mathbb{P}(X)$ and a mechanism that samples from $\mathbb{P}(X)$.

## A.2 Background on algorithmic causality

The formalism of algorithmic causality derives from [15, 28]. Here we directly restate or paraphrase additional relevant definitions, postulates, and theorems from [15].

### A.2.1 Algorithmic information theory

The following definition precisely captures what loosely can be thought of as the amount of information contained in an object (represented by a string):

**Definition A.1 (Kolmogorov-complexity).** *Given the set of binary strings of arbitrary length $\{0,1\}^*$, the **Kolmogorov complexity** $K(s)$ of a string $s \in \{0,1\}^*$ is the length of the shortest program that generates $s$ using a previously defined universal Turing machine [7, 8, 17, 19, 31, 32]. The **conditional Kolmogorov complexity** $K(x|y)$ of a string $x$ given another string $y$ is the length of the shortest program that generates $x$ from the shortest description of $y$.*

The following definitions derive algorithmic mutual information from Kolomogov complexity and algorithmic independence from algorithmic mutual information:

**Definition A.2 (algorithmic mutual information).** *The **algorithmic mutual information** of two binary strings $x, y$ is*

$$I(x : y) :\stackrel{+}{=} K(x) + K(y) - K(x, y).$$

**Definition A.3 (algorithmic independence).** *Given three strings $x, y, z$, $x$ is **algorithmically conditionally independent** of $y$, given $z$ if the additional knowledge of $y$ does not allow for stronger compression of $x$, given $z$:*

$$x \perp\!\!\!\perp y|z \;\Leftrightarrow\; I(x : y|z) \stackrel{+}{=} 0.$$

### A.2.2 Causality

The following definition is the probabilistic analog of the algorithmic causal model in Thm. 3.1:

**Definition A.4 (structural-causal-model).** *A **structural causal model** (SCM) [26, 27] represents the assignment of random variable $X$ as the output of a function, denoted by lowercase monospace (e.g. f), that takes as input an independent noise variable $N_X$ and the random variables $\{PA_X\}$ that represent the parents of $X$ in a directed acyclic graph (DAG) $\mathcal{G}$:*

$$X := \mathtt{f}(\{PA_X\}, N_X). \tag{3}$$

*Given the noise distributions $\mathbb{P}(N_X)$ for all variables $X$ in $\mathcal{G}$, SCM entails a joint distribution $\mathbb{P}$ over all the variables in $\mathcal{G}$ [28].*

The graph-theoretic concept of $d$-separation is used for determining conditional independencies induced by a directed acyclic graph (see point 3 in Thm. A.1):

**Definition A.5** ($d$**-separation**). *A path $p$ in a $DAG$ is said to be d-separated (or blocked) by a set of nodes $Z$ if and only if*

1. *$p$ contains a chain $i \to m \to j$ or fork $i \leftarrow m \to j$ such that the middle node $m$ is in $Z$, or*
2. *$p$ contains an inverted fork (or collider) $i \to m \leftarrow j$ such that the middle node $m$ is not in $Z$ and such that no descendant of $m$ is in $Z$.*

*A set of nodes $Z$ d-**separates** a set of nodes $X$ from a set of nodes $Y$ if and only if $Z$ blocks every (possibly undirected) path from a node in $X$ to a node in $Y$.*

### A.2.3 Algorithmic causality

We now recall key results from past work stated in §3.1. The algorithmic model of causality formalizes the graphical structure of the programs considered in this paper:

**Definition 3.1 (algorithmic model of causality).** *Let $\mathcal{G}$ be a directed acyclic graph (DAG) formalizing the causal structure among the strings $x_1, ..., x_N$. Every $x_j$ is computed by a program $\mathtt{F}_j$ with length $O(1)$ from its parents $\{pa_j\}$ and additional noise input $n_j$. Formally,*

$$x_j := \mathtt{F}_j(\{pa_j, n_j\}),$$

*meaning that the Turing machine computes $x_j$ from the input $\{pa_j\}, n_j$ using the additional program $\mathtt{F}_j$ and halts. Assume the noise $n_j$ are jointly independent in the sense*

$$n_j \perp\!\!\!\perp \{n_{\neq j}\}.$$

Post. 3.2 states our assumption that all algorithmic independencies are accounted for in the causal structure, a property known as faithfulness [33]:

**Postulate 3.2 (faithfulness).** *Consider the DAG $\mathcal{G}$ from Def. 3.1. Given three sets $S$, $T$, $R$ of nodes, $I(S : T|R) \stackrel{+}{=} 0$ implies $R$ d-separates $S$ and $T$.*

Thms. 3.1 and A.1 establish a connection between the graph-theoretic concept of $d$-separation with condition algorithmic independence of the nodes of the graph.

**Theorem 3.1 (algorithmic causal Markov condition).** *Consider the DAG $\mathcal{G}$ from Def. 3.1. Let $\{nd_j\}$ be the concatenation of all non-descendants of $x_j$ except itself. Then, $x_1, ..., x_N$ satisfy $x_j \perp\!\!\!\perp \{nd_j\}|\{pa_j\}$.*

**Theorem A.1 (equivalence of algorithmic Markov conditions).** *Given the strings $x_1, ..., x_n$ and a directed acyclic graph $\mathcal{G}$, the following conditions are equivalent:*

1. ***Recursive form:** the joint complexity is given by the sum of complexities of each node $x_j$, given the optimal compression of its parents $\{pa_j\}$:*

$$K(x_1, ..., x_n) \stackrel{+}{=} \sum_{j=1}^{n} K(x_j|\{pa_j\}).$$

2. ***Local Markov Condition:** Every node $x_j$ is independent of its non-descendants $\{nd_j\}$, given the optimal compression of its parents $\{pa_j\}$:*

$$I(x_j : nd_j|\{pa_j\}) \stackrel{+}{=} 0.$$

3. ***Global Markov Condition:** Given three sets $S$, $T$, $R$ of nodes*

$$I(S : T|R) \stackrel{+}{=} 0$$

*if $R$ d-separates $S$ and $T$.*

Together, Thms. 3.1, A.1 and Post. 3.2 establish our assumptions that a conditional independence exists if and only if it is reflected in the structure of the directed acyclic graph computed via programs. This is as reasonable of an assumption as assuming that the real RL algorithms we consider in this paper are written in a way that is domain agnostic, which is a common working assumption for programming general-purpose RL algorithms.

## B Proofs

This section contains the proofs for our theoretical results.

**Proposition 5.1 (algorithmic causal model of RL).** *Let $\mathcal{G}^{RL}$ specify the complete algorithmic graphical structure of any reinforcement learning process. The nodes of $\mathcal{G}^{RL}$ are the variables $(s, w_s, n_s, b_s, s', r)$ of all decision events during learning. These nodes are computed via algorithmically independent programs with length $O(1)$ that constitute the subprograms that implement* `A` *and* `U`*. Then the nodes satisfy the algorithmic causal Markov condition under $\mathcal{G}^{RL}$.*

*Proof.* $\mathcal{G}^{\text{RL}}$ is defined according to Def. 3.1. Therefore Thm. 3.1 applies to $\mathcal{G}^{\text{RL}}$. □

**Proposition 6.1 (independence ⇒ structural locality).** *If an agent architecture is not* ***structurally local****, then its decision mechanisms are never mutually independent conditioned on its causal ancestors in $\mathcal{G}^{RL}$.*

*Proof.* Consider two decision mechanisms $\mathtt{D}^i$ and $\mathtt{D}^j$, without loss of generality. $\mathtt{D}^i$ is encoded by weights $w^i = [w^a, w^c]$ and $\mathtt{D}^j$ is encoded by weights $w^j = [w^b, w^c]$, where we have assumed that $w^a, w^b, w^c$ are initialized to be jointly independent. Our working assumption is that the weights of any decision mechanism have non-negligible complexity from their random initialization, and that they maintain non-negligible complexity throughout learning. We prove, via induction on the number of credit assignment updates $n$, that $\mathtt{D}^i$ and $\mathtt{D}^j$ are never independent at any step of the learning process.

**Base case:** $n = 0$. *Before the first credit assignment update, $\mathtt{D}^i$ and $\mathtt{D}^j$ are not unconditionally independent.*

All weights are initialized independently, so it suffices for us to show that $\mathtt{D}^i$ and $\mathtt{D}^j$ are unconditionally independent because the initial weights do not have causal ancestors in $\mathcal{G}^{\text{RL}}$. We observe that

$$K(\mathtt{D}^i) \stackrel{+}{=} K(w^a, w^c) \stackrel{+}{=} K(w^a) + K(w^c)$$
$$K(\mathtt{D}^j) \stackrel{+}{=} K(w^b, w^c) \stackrel{+}{=} K(w^b) + K(w^c)$$
$$K(\mathtt{D}^i, \mathtt{D}^j) \stackrel{+}{=} K(w^a, w^b, w^c) \stackrel{+}{=} K(w^a) + K(w^b) + K(w^c),$$

where we have assumed the description complexity of how weights produces bids from states is constant. Then

$$I(\mathtt{D}^i : \mathtt{D}^j) \stackrel{+}{=} K(\mathtt{D}^i) + K(\mathtt{D}^j) - K(\mathtt{D}^i, \mathtt{D}^j) \stackrel{+}{=} K(w^c).$$

Given our assumption that weights at any point during the learning process have non-negligible complexity, in particular that $K(w^c) \stackrel{+}{\gg} 0$, the mutual information between $\mathtt{D}^i$ and $\mathtt{D}^j$, and therefore $w^{i\prime}$ and $w^{j\prime}$, is non-negligible. Thus $\mathtt{D}^i$ and $\mathtt{D}^j$ are not independent at the beginning of the learning process.

**Inductive hypothesis:** *Assuming that $\mathtt{D}^i$ and $\mathtt{D}^j$ are not independent conditional on other nodes in $\mathcal{G}^{RL}$ after $n$ iterations, $\mathtt{D}^i$ and $\mathtt{D}^j$ are not independent after $n + 1$ iterations.*

**Inductive step:** We now show that the updated weights of $\mathtt{D}^i$ will have nonzero mutual information with the updated weights of $\mathtt{D}^j$ after a credit assignment update. We can model the credit assignment mechanism as modifying the contents of three separate memory locations $a$, $b$, and $c$, which store the contents of the updated versions of $w^a, w^b, w^c$, after which the pointers to the encodings of $\mathtt{D}^i$ and $\mathtt{D}^j$ are updated to point to these memory locations. The updated version of the shared weights $w^c$ resides in a shared memory location that both $\mathtt{D}^i$ and $\mathtt{D}^j$ reference. Assume without loss of generality that $\mathtt{D}^i$ is updated before $\mathtt{D}^j$ (since the Turing machine operates serially). Updating $\mathtt{D}^i$ updates $w^i$ to $w^{i\prime}$ updates $w^a$ to $w^{a\prime}$ and $w^c$ to $w^{c\prime}$.

| Memory Location | $a$ | $b$ | $c$ |
|---|---|---|---|
| Contents | $w^{a\prime}$ | N/A | $w^{c\prime}$ |

Then updating $\mathtt{D}^j$ updates $w^j$ to $w^{j\prime}$ updates $w^b$ to $w^{b\prime}$ and overwrites $w^{c\prime}$ to $w^{c\prime\prime}$.

| Memory Location | $a$ | $b$ | $c$ |
|---|---|---|---|
| Contents | $w^{a\prime}$ | $w^{b\prime}$ | $w^{c\prime\prime}$ |

Thus after the credit assignment update, we have

$$K(\mathtt{D}^{i\prime}) \stackrel{+}{=} K(w^{a\prime}, w^{c\prime\prime}) \stackrel{+}{=} K(w^{a\prime}) + K(w^{c\prime\prime})$$
$$K(\mathtt{D}^{j\prime}) \stackrel{+}{=} K(w^{b\prime}, w^{c\prime\prime}) \stackrel{+}{=} K(w^{b\prime}) + K(w^{c\prime\prime})$$

so by the same reasoning as in the base case we have

$$I(\mathtt{D}^{i\prime} : \mathtt{D}^{j\prime}) \stackrel{+}{=} K(\mathtt{D}^{i\prime}) + K(\mathtt{D}^{j\prime}) - K(\mathtt{D}^{i\prime}, \mathtt{D}^{j\prime}) \stackrel{+}{=} K(w^{c\prime\prime}).$$

Given our assumption that weights at any point during the learning process have non-negligible complexity, in particular that $K(w^{c\prime\prime}) \stackrel{+}{\gg} 0$, the mutual information between $\mathtt{D}^{i\prime}$ and $\mathtt{D}^{j\prime}$, and therefore $w^{i\prime}$ and $w^{j\prime}$, is non-negligible after a credit assignment update. Inspecting the graph of $\mathcal{G}^{RL}$ shows that this lack of independence holds true conditioned on any other nodes $s, w_s, n_s, b_s, s', r$ anywhere in $\mathcal{G}^{RL}$.

*Remark.* The assumption that initialized weights have non-negligible complexity during random initialization is as reasonable as assuming that the random string produced by the computer's random number generator is incompressible, which is a standard assumption at the level of analysis of RL research. This paper assumes that weights maintain their non-negligible complexity throughout learning, but exogenous supervision signal that regularize the complexity of the weights could potentially also decrease the mutual information among decision mechanisms. Understanding the effect of such regularization techniques is out of the scope of this paper, but would be interesting for future work.

□

**Theorem 6.2 (modularity).** *An RL algorithm is modular if and only if its agent architecture is structurally local and credit assignment mechanism is temporally local.*

*Proof.* ⇐ **direction:** We first prove that if the decision mechanisms do not share weights of non-negligible complexity and the input to the credit assignment mechanism $\mathtt{U}$ for updating the active decision mechanism $\mathtt{D}^i$ at $e_t$ is a subset of the information available at $e_t$, then for all $w^{j'}_{s_{\tilde{t}}}$ where $\tilde{t} \neq t$ and $j \neq i$,

$$I\left(w^{i\prime}_{s_t} : w^{j\prime}_{s_{\tilde{t}}} \middle| s_t, s_{t+1}, w\right) \stackrel{+}{=} 0.$$

By Prop. 5.1 and Thm. A.1, this reduces to showing that for all $w^{j'}_{s_{\tilde{t}}}$ where $\tilde{t} \neq t$ and $j \neq i$, $w^{i\prime}_{s_t}$ and $w^{j\prime}_{s_{\tilde{t}}}$ are $d$-separated by $s_t, s_{t+1}, w$. Structural locality means that there is no direct connection between $w^{i\prime}_{s_t}$ and $w^{j\prime}_{s_{\tilde{t}}}$ through a shared subset of weights: all paths between $w^{i\prime}_{s_t}$ and $w^{j\prime}_{s_{\tilde{t}}}$ must go through other nodes in $\mathcal{G}^{\text{RL}}$. $w'$ is the latest node to be generated at this point in the computational process, so we need only consider paths coming through the causal ancestors of $w^{i\prime}_{s_t}$ and $w^{j\prime}_{s_{\tilde{t}}}$ rather than their descendants, which have not been generated yet. This means that the only paths into $w^{i\prime}_{s_t}$ must come directly through the nodes that represent the input arguments of $\mathtt{U}$ applied to $\mathtt{D}^i$. If $\mathtt{U}$ is temporally local, then these input arguments are restricted to $s_t, w, n_{s_t}, b_{s_t}, s_{t+1}, r_t$. Having conditioned on $s_t, s_{t+1}, w$, what is left to show is that $s_t, s_{t+1}, w$ blocks all paths between $w^{i\prime}_{s_t}$ and $w^{j\prime}_{s_{\tilde{t}}}$ through $n_{s_t}, b_{s_t}, r_t$. Indeed this is the case, as shown in Fig. 1. We observe directly that $s_t$ is the common causal ancestor of $n_{s_t}, b_{s_t}, r_t$ with no edges to $w^{j\prime}_{s_{\tilde{t}}}$ in between $s_t$ and $n_{s_t}, b_{s_t}, r_t$.

⇒ **direction:** Next we prove that

$$I\left(w^{i\prime}_{s_t} : w^{j\prime}_{s_{\tilde{t}}} \middle| s_t, s_{t+1}, w\right) \stackrel{+}{=} 0.$$

implies structural locality and temporal locality. The contrapositive of Prop. 6.1 states that structural locality is implied by modularity.

Next we consider temporal locality. First, recall that we have restricted our scope of analysis to action-value, policy gradient, and actor-critic methods, all of which use information at least from $e_t$

to update the active decision mechanism at time-step $t$. Recall also by definition of $d$-separation (case 1 in Def. A.5), if two causal mechanisms share the same input, the output of the causal mechanisms are not $d$-separated if this input is not in the set being conditioned on.

We want to show that $I(w^{i\prime}_{s_t} : w^{j\prime}_{s_{\tilde{t}}} | s_t, s_{t+1}, w) \overset{+}{=} 0$ implies that the inputs to U are confined to $e_t$. Now assume for the sake of contradiction that the inputs to U are not confined to $e_t$; that is, the inputs to U contain information from $e_{\tilde{t}}$ other than $s_t, s_{t+1}, w$ for some other $\tilde{t} \neq t$ in the same decision sequence $\bar{e}$. Without loss of generality let $w^{j\prime}_{s_{\tilde{t}}}$ be the active decision mechanism at $e_{\tilde{t}}$. Then $w^{i\prime}_{s_t}$ and $w^{j\prime}_{s_{\tilde{t}}}$ are not $d$-separated by $s_t, s_{t+1}, w$. Then by Post. 3.2, $w^{i\prime}_{s_t}$ and $w^{j\prime}_{s_{\tilde{t}}}$ would not be mutually independent given $s_t, s_{t+1}, w$. □