Modularity in Reinforcement Learning via Algorithmic Independence in Credit Assignment

Michael Chang* [1] **Sidhant Kaushik*** [1] **Sergey Levine** [1] **Thomas L. Griffiths** [2]

Abstract

Many transfer problems require re-using previously optimal decisions for solving new tasks, which suggests the need for learning algorithms that can modify the mechanisms for choosing certain actions independently of those for choosing others. However, there is currently no formalism nor theory for how to achieve this kind of modular credit assignment. To answer this question, we define modular credit assignment as a constraint on minimizing the algorithmic mutual information among feedback signals for different decisions. We introduce what we call the modularity criterion for testing whether a learning algorithm satisfies this constraint by performing causal analysis on the algorithm itself. We generalize the recently proposed societal decision-making framework as a more granular formalism than the Markov decision process to prove that for decision sequences that do not contain cycles, certain single-step temporal difference action-value methods meet this criterion while all policy-gradient methods do not. Empirical evidence suggests that such action-value methods are more sample efficient than policy-gradient methods on transfer problems that require only sparse changes to a sequence of previously optimal decisions.

> It is causality that gives us this modularity, and when we lose causality, we lose modularity.
>
> Judea Pearl (Ford, 2018)

*Equal contribution [1]Department of Computer Science, University of California, Berkeley, USA [2]Department of Computer Science, Princeton University, USA. Correspondence to: Michael Chang <mbchang@berkeley.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1. Introduction

Gusteau's (Bird et al., 2007) taqueria has a great team for making burritos: Colette heats the tortillas, Remy adds the meat, and Alfredo wraps the burrito in aluminum foil. But today customers fell sick from meat contamination and gave angry reviews. Clearly, Remy should replace meat with tofu or something else. But should credit assignment from the reviews affect the others? Intuitively, no: the feedback signals to the decision of adding meat and to the decisions of heating tortillas and wrapping aluminum foil should be independent. Customer dissatisfaction in burritos are not reflective of the taqueria's quesadillas, for which Colette's tortilla skills and Alfredo's wrapping skills are useful.

The example above expresses the intuition that **modularity**, or the capacity for the mechanisms in a system to be independently modified, enables flexible adaptation. However, using principles of modularity to build flexible learning systems has been difficult because the traditional formalism for precisely describing what modularity means was developed in the context of analyzing **static systems** – systems whose mechanisms are assumed fixed. But learning agents are **dynamic systems** composed of mechanisms (i.e. learnable functions) that evolve over the course of learning. Thus, to even express the hypothesis that modularity enables flexibility in learning agents, let alone test it, we first need (1) a formalism that defines what modularity means for dynamic

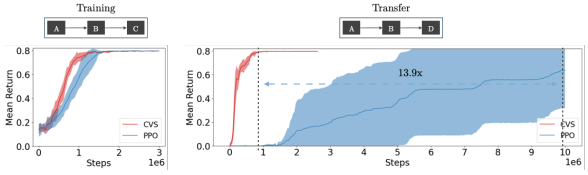

Figure 1: **Minimal motivating example.** The optimal action sequence for the training task is $A \rightarrow B \rightarrow C$, and the optimal sequence for the transfer task differs only in the last time-step. Continuing to train an optimal policy from the training task on the transfer task with the cloned Vickrey society (CVS) from Chang et al. (2020) transfers 13.9x more efficiently than with PPO (Schulman et al., 2017), even though learning efficiency for both on-policy algorithms during training is comparable. This paper suggests that this is due to **dynamic modularity**: the algorithmic independence among CVS's learnable mechanisms and among their gradients.

systems, (2) a theory that identifies the conditions under which independent modification of learnable mechanisms is even possible, and (3) a practical criterion for determining when these conditions are met in learning algorithms. This paper proposes candidate answers to these questions and applies them to shed new insight on the modularity of discrete-action reinforcement learning (RL) algorithms. The takeaway message of this paper is that independent modification of mechanisms requires both the mechanisms *and* the feedback signals that update them to be independent.

Janzing & Schölkopf (2010) proposed to precisely characterize modularity in static systems as the algorithmic independence of mechanisms in the computational graph used to describe the system. In learning systems, the computational graph in question depicts the forward pass of a learner (e.g. a neural network), but this graph itself evolves over the course of learning because the learnable functions – the mechanisms – get modified. For such dynamic systems, we extend the static notion of modularity to define **dynamic modularity** as the algorithmic independence of mechanisms in the current iteration, conditioned on the graph from the previous iteration of evolution. This addresses question (1).

Modularity matters when the system needs to be modified for a new context or purpose. In learning systems it is the credit assignment mechanism that performs this modification. Thus dynamic modularity is tied to independence in feedback: for a gradient-based learner, we show that enforcing dynamic modularity requires enforcing gradients to be algorithmically independent as well, which we call the **modularity constraint.** This addresses question (2).

Algorithmic independence is generally incomputable, which makes the modularity constraint intractable to evaluate. To make the constraint practical for analyzing learning algorithms, we reconcile the computational graph of execution, which evaluates learnable mechanisms as functions, with the computational graph of credit assignment, which mutates them as data, to represent the entire learning process as one big causal graph, which we call the **algorithmic causal model of learning** (ACML). Then the modularity constraint translates into an easy-to-inspect criterion, the **modularity criterion**, on d-separation in ACML that enables us to evaluate, without any training, whether a learning algorithm is exhibits dynamic modularity. This addresses question (3).

Having established a theoretically-grounded formalism for reasoning about modularity in learning systems, we theoretically and empirically analyze discrete-action RL algorithms. The mechanisms of interest are the functions that compute the "bid" (e.g. action probability or Q-value) for each value of the action variable. The Markov decision process (MDP) is too coarse-grained to represent these functions separately, so we use the societal decision-making framework (SDM) from Chang et al. (2020), whose computational graph does treat them separately. We prove that certain single-step temporal difference methods satisfy the modularity criterion while all policy gradient methods do not. Empirically, we find that for transfer problems that require only sparse modifications to a sequence of previously optimal decisions, implementations of algorithms that exhibit dynamic modularity transfer more efficiently than their counterparts. All proofs are in the Appendix.

2. Related Work

The hypothesis that modularity could improve flexibility of learning systems has motivated much empirical work in designing factorized architectures (Devin et al., 2017; Andreas et al., 2016; Chang et al., 2018; Goyal et al., 2019; Kirsch et al., 2018; Alet et al., 2018; Pathak et al., 2019) and reinforcement learners (Simpkins & Isbell, 2019; Sprague & Ballard, 2003; Samejima et al., 2003), but the extent to which the heuristics used in these methods enforce the learnable components to be independently modifiable has yet to be tested. Conversely, other works begin by defining a multi-agent system of independently modifiable components and seek methods to induce their cooperation against a global objective (Balduzzi, 2014; Baum, 1996; Srivastava et al., 2013; Chang et al., 2020; Gemp et al., 2020; Balduzzi et al., 2020), but the precise property of a learning system that characterizes its modularity has not been discussed in these works, as far as we are aware. Recent complementary work has proposed alternative measures of modularity, restricted to deep networks, based on connectivity strength (Filan et al., 2021) and functional decomposition (Csordás et al., 2020). In contrast, our work identifies a general property that defines the modularity of a learning system as the algorithmic independence of learnable mechanisms and of their gradients, and presents a practical method for testing for this property without any training. We build upon the theoretical foundations from Janzing & Schölkopf (2010) that have clarified similar notions of "autonomy" and "invariance" that underlie axioms of econometrics (Haavelmo, 1944; Aldrich, 1989), causality (Pearl, 2009; Peters et al., 2017), and computer programming (Abelson & Sussman, 1996). Yu et al. (2020) explored enforcing the linear independence of gradients to improve multi-task learning, and formulating the precise connection between algorithmic and linear independence would be valuable future work.

3. Background

Our analysis of the modularity of RL algorithms employs two key ideas: (§3.1) computational graphs can be interpreted as causal graphs and (§3.2) a learnable discrete-action policy can be interpreted as a society of learnable action-specific functions and a fixed selection mechanism.

3.1. Algorithmic Causality

We begin by reviewing terms from algorithmic information theory (Kolmogorov, 1965; Li et al., 2008; Solomonoff, 1964)[1]. We assume that programs are expressed in a language L and run on a universal Turing machine. Given binary strings x, y, and z, we denote **conditional algorithmic independence** as $x \perp\!\!\!\perp y \mid z$, equivalently $I(x : y \mid z) \stackrel{+}{=} 0$, which reads "given z, knowledge of y does not allow for a stronger compression of x." I denotes **conditional algorithmic mutual information**. $\stackrel{+}{=}$ denotes equality up to a constant that depends on L but not the strings on either side of the equality. **Conditional Kolmogorov complexity** of y given x is given by $K(y \mid x)$, the length of the shortest program that generates y from x as input.

Janzing & Schölkopf (2010, Post. 6) generalized structural causal models (Pearl, 1995) to general programs, allowing us to treat computational graphs as causal graphs.

Definition 1 (**computational graph**). *Define a* ***computational graph*** $\mathtt{G} = (\boldsymbol{x}, \boldsymbol{f})$ *as a directed acyclic factor graph (DAG) of variable nodes* $\boldsymbol{x} = x_1, ..., x_N$ *and function nodes* $\boldsymbol{f} = f^1, ..., f^N$. *Let each* x_j *be computed by a program* f^j *with length* $O(1)$ *from its parents* $\{pa_j\}$ *and possibly an additional noise input* n_j. *Assume the noise* n_j *are jointly independent:* $n_j \perp\!\!\!\perp \{n_{\neq j}\}$. *Formally,* $x_j := f^j(\{pa_j\}, n_j)$, *meaning that the Turing machine computes* x_j *from the input* $\{pa_j\}, n_j$ *using the additional program* f^j *and halts.*

This DAG represents a probabilistic program (van de Meent et al., 2018; Goodman et al., 2016; Mansinghka et al., 2009) in the general case, and either a standard causal model if some $\mathtt{f}_j$ takes in a noise variable or a deterministic program if none do. Henceforth we treat all graphs as computational graphs. We define a **mechanism** as the string representation (i.e. source code) of the program that implements a function $\mathtt{f}$ and **data** as the string representations of the input/output variables x of $\mathtt{f}$. The **algorithmic causal Markov condition** (Janzing & Schölkopf, 2010, Thm. 4), which states that d-separation implies conditional independence, generalizes the standard Markov condition to general programs:

Theorem 1 (**algorithmic causal Markov condition**). *Let* $\{pa_j\}$ *and* $\{nd_j\}$ *respectively represent concatenation of the parents and non-descendants (except itself) of* x_j *in a computational graph. Then* $\forall x_j$, $x_j \perp\!\!\!\perp \{nd_j\} \mid \{pa_j\}$.

In standard causality it is typical to assume the converse of the Markov condition, known as **faithfulness** (Spirtes et al., 2000). We do the same for algorithmic causality:

Postulate 2 (**algorithmic faithfulness**). *Given sets* S, T, R *of nodes in a computational graph,* $I(S : T|R) \stackrel{+}{=} 0$ *implies* R *d-separates* S *and* T.

[1] See the appendix for background.

3.2. Societal Decision-Making

A discrete-action MDP is the standard graph for a sequential decision problem over states S with N discrete actions A, defined with state space $\mathcal{S}$, action space $\{1, ..., N\}$, transition function $\mathtt{T} : \mathcal{S} \times \{1, ..., N\} \to \mathcal{S}$, reward function $\mathtt{R} : \mathcal{S} \times \{1, ..., N\} \to \mathbb{R}$, and discount factor γ. The MDP objective is to maximize the return $\sum_{t=0}^{T} \gamma^t \mathtt{R}(s_t, a_t)$ with respect to a policy $\pi : \mathcal{S} \to \{1, ..., N\}$. We define a **decision** as a value a of A. The MDP abstracts over the mechanisms that control each decision with a single edge in the graph, represented by π, but to analyze the independence of different decisions we are interested in representing these mechanisms as separate edges.

The societal decision-making (SDM) framework (Chang et al., 2020) offers an alternative graph that does exactly this: it decomposes a discrete-action policy as a society of N agents ω^k that each controls a different decision. Each agent is a tuple (ψ^k, ϕ^k) of a bidder $\psi^k : \mathcal{S} \to \mathcal{B}$ and a fixed transformation $\phi^k : \mathcal{S} \to \mathcal{S}$. In §6, we will consider the algorithmic independence of the ψ^n. Recovering a policy π composes two operations: one computes bids $b_s^k := \psi^k(s),\ \forall k$, and one applies a selection mechanism $\mathtt{S} : \mathcal{B}^N \to \{1, ..., N\}$ on the bids to select decision a. SDM thus curries the transition and reward functions as $\mathtt{T} : \{1, ..., N\} \to [\mathcal{S} \to \mathcal{S}]$ and $\mathtt{R} : \{1, ..., N\} \to [\mathcal{S} \to \mathbb{R}]$.

Chang et al. (2020) introduced the cloned Vickrey society (CVS) algorithm as an on-policy single-step temporal-difference action-value method. CVS interprets the Bellman optimality equation as an economic transaction between agents seeking to optimize their utilites in a Vickrey auction (Vickrey, 1961) at each time-step. The Vickrey auction is the selection mechanism that selects the highest bidding agent i, which receives a utility

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

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 function parameters. The 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. Dynamic Modularity in Learning Systems

This section extends the definition of modularity in static systems to dynamic systems. We discuss learning algorithms as examples of dynamic systems and the constraints that must be imposed on the credit assignment mechanism for the learning algorithm to be exhibit dynamic modularity.

4.1. From Static Modularity to Dynamic Modularity

A system can be described by a computational graph, whose mechanisms represent the system components (e.g. stars of a star system) and whose data represent the information communicated between components (e.g. forces between the stars). We describe a **dynamic system** with a **dynamic graph** whose mechanisms symmetrically evolve through time (e.g. the stars move), processed by a meta-mechanism that is equivariant to re-indexing of the mechanisms (e.g. physical laws governing stars' motion are invariant to change reference frame). In this sense, mechanisms in dynamic graphs can be treated as not only functions but also variables (with respect to the meta-mechanism), which we will exploit. A **static system**, described by a **static graph** with fixed mechanisms, represents a snapshot in time of a dynamic system. Treating mechanisms as fixed is often assumed in standard causal analysis (Pearl, 2009). Modularity in the static context has been defined as the autonomy (Pearl, 2009, §1.3.1), or more precisely, the algorithmic independence (Janzing & Schölkopf, 2010, Post. 7) of mechanisms:

Definition 2 (**static modularity**).

$$\forall k \neq j, \quad I\left(\mathtt{f}^k : \mathtt{f}^j\right) \stackrel{+}{=} 0. \tag{2}$$

We now add a temporal dimension i to define modularity for dynamic systems, which we assume are Markovian:

Definition 3 (**dynamic modularity**).

$$\forall k \neq j, \quad I\left(\mathtt{f}^{k,i+1} : \mathtt{f}^{j,i+1} \mid \boldsymbol{x}^i, \boldsymbol{\mathtt{f}}^i\right) \stackrel{+}{=} 0. \tag{3}$$

Dynamically modularity essentially considers the static modularity of a system at a particular snapshot in time, treating the past iterations as background information.

4.2. Learning Algorithms are Dynamic Systems

We now show that general learning algorithms are examples of dynamic systems and can be analyzed as such. To do so, we need to specify the data and mechanisms of the static computational graph that represents a particular snapshot, as well as the equivariant meta-mechanism that evolves the mechanisms from one iteration to the next.

Let the **model of execution** be the computational graph $\mathtt{E}$ that represents the forward pass of the learner, generating $\mathbf{x}$ as an execution trace $(x_1, ..., x_t, ..., x_T)$ of the input and output data of the learnable mechanisms $\mathtt{f}$. For example, with MDPs, the forward pass is a rollout, the trace records its states, actions, and rewards, and the mechanisms, which map parent variables $\{pa\}_t = s_t$ to child variables $x_t = (a_t, s_{t+1}, r_t)$, are instances of the policy at different steps t.

Let the **model of credit assignment** be the computational graph $\mathtt{C}$ that evolves the mechanisms. Each step represents the backward pass of the learner. Here the mechanisms are treated as data for two equivariant meta-mechanisms, the credit assignment mechanism $\Pi(\mathbf{x}, \mathtt{f}) \to \boldsymbol{\delta}$ and the update rule $\texttt{UPDATE}(\mathtt{f}, \boldsymbol{\delta}) \to \mathtt{f}'$. $\mathtt{C}$ can be viewed as a rewardless MDP with states $\mathtt{f}$ and actions $\boldsymbol{\delta}$, with UPDATE as the transition function. Then Π is a context-conditioned policy that generates modifications $\boldsymbol{\delta} = (\delta_1, ..., \delta_T)$ to the functions $\mathtt{f}$ of the learner, given $\mathbf{x}$ as context. For a gradient-based learner, δ_t^k is the gradient of the learning objective with respect to the function $\mathtt{f}^k$ that participated at step t of the execution trace (e.g. as we discuss in §6, δ_t^k would be the Bellman error of the decision mechanism for action k taken at step i). UPDATE performs the parallel operation $\texttt{UPDATE}(\mathtt{f}^k, \sum_t \delta_t^k) \to \mathtt{f}^{k\prime}$ over all mechanisms $\mathtt{f}^k$. The choice of optimizer for gradient descent (e.g. Adam (Kingma & Ba, 2014)) determines the functional form of UPDATE. Henceforth we assume gradient-based learning, but our results hold more generally given the assumptions that UPDATE (1) is algorithmically independent of Π and (2) completely factorizes across k.

4.3. Modularity Constraint on Credit Assignment

The design of a learning algorithm primarily concerns the credit assignment mechanism Π, whereas the choice of UPDATE is often assumed. We now present the constraint Π must satisfy for dynamic modularity to hold at every iteration of learning. Given trace $\mathbf{x}$ and previous mechanisms $\mathtt{f}$, we define the **modularity constraint** as that which imposes that the gradients $\delta_1, ..., \delta_T$ be jointly independent:

Definition 4 (**modularity constraint**).

$$I\left(\delta_1, ..., \delta_T \mid \boldsymbol{x}, \boldsymbol{\mathtt{f}}\right) \stackrel{+}{=} 0. \tag{4}$$

A **modular credit assignment mechanism** is one that satisfies the modularity constraint. If $\mathtt{E}$ exhibited statically modularity (i.e. its functions were independently initialized) then a modular Π enforces dynamic modularity:

Theorem 3 (**modular credit assignment**). *Dynamic modularity is enforced at learning iteration i if and only if static modularity holds at iteration $i = 0$ and the credit assignment mechanism satisfies the modularity constraint.*

Initializing different functions with different weights is not sufficient to guarantee dynamic modularity. The gradients produced by Π must be independent as well. If Π were not modular it would be impossible for it to modify a function without simultaneously inducing a dependence with another, other than via non-generic instances where δ_t has a simple description, i.e. $\delta_t = 0$, which, unless imposed, are unlikely to hold over all iterations of learning.

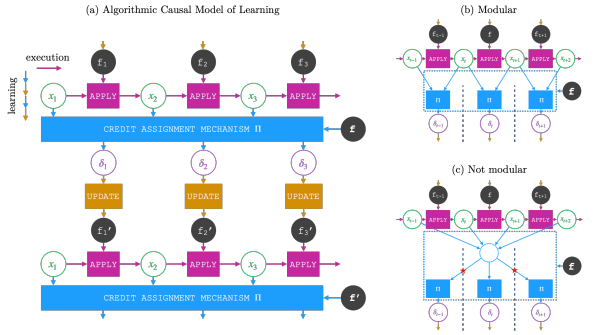

Figure 2: **Algorithmic Causal Model of Learning.** A learning algorithm with credit assignment mechanism Π that produces gradients δ to update functions $\mathtt{f}$ to $\mathtt{f}'$ can be represented as a causal graph (a). Π is not modular (b) if it contains a hidden variable whose outgoing causal edges cut a partitioning among the δ's (shown by the red star) and modular (c) if it does not.

5. An Algorithmic Causal Model of Learning

We can determine the dynamic modularity of a learning algorithm if we can evaluate the modularity constraint, but evaluating it is not practical in its current form because algorithmic information is generally incomputable. This section proposes to bypass this incomputability by translating the constraint into a d-separation criterion on the causal structure of Π, defined as part of one single causal graph of the learning process, which combines both the model of execution and the model of credit assignment. The challenge to constructing this graph is that $\mathbf{f}$ are treated as functions in $\mathbf{E}$ but as data in $\mathbf{C}$, so it is not obvious how to reconcile the two in the same graph. We solve this by treating the function application operation $\mathtt{APPLY}$ (Abelson & Sussman, 1996)[2], where $\forall \mathtt{f}, x,\ \mathtt{APPLY}(\mathtt{f}, x) := \mathtt{f}(x)$, as itself a function in a computational graph, enabling us to treat both $\mathtt{f}$ and x as variables in the same flattened dynamic graph (Fig. 2a).

Lemma 4 (**algorithmic causal model of learning**). *Given a model of execution $\mathbf{E}$ and of credit assignment $\mathbf{C}$, define the **algorithmic causal model of learning** (ACML) as a dynamic computational graph $\mathbb{L}$ of the learning process. We assume Π has its own internal causal structure with internal variable and function nodes. The function nodes of $\mathbb{L}$ are $\mathtt{APPLY}$, $\mathtt{UPDATE}$, and the internal function nodes of Π, all with length $O(1)$. The variable nodes of $\mathbb{L}$ are x, $\mathtt{f}$, δ, and internal variable nodes of Π. Then these variable nodes satisfy the algorithmic causal Markov condition with respect to $\mathbb{L}$ for all steps of credit assignment.*

ACML is the bridge that brings tools from algorithmic causality (Janzing & Schölkopf, 2010) to bear on analyzing not simply the algorithmic independence of variables, but algorithmic independence of *functions* in general learning algorithms. The learnable mechanisms are no longer considered to have length $O(1)$ as is assumed in the model of execution. With ACML, we define a criterion to test whether the modularity constraint holds by direct inspection:

Theorem 5 (**modularity criterion**). *If $\mathbb{L}$ is faithful, the modularity constraint holds if and only if for all i, outputs δ_t and $\delta_{\neq t}$ of Π are d-separated by its inputs $\mathbf{x}$ and $\mathbf{f}$.*

We generally have access to the true computational graph, because the learning algorithm was programmed by us. Thus Thm. 5 enables us to evaluate, before any training, whether a learning algorithm satisfies the modularity constraint by simply inspecting $\mathbb{L}$ for d-separation (Fig. 2b,c), giving us a practical tool to both design and evaluate learning algorithms on the basis of dynamic modularity.

6. Modularity in Reinforcement Learning

We now apply the modularity criterion to evaluate the dynamic modularity of two major classes of RL algorithms (Sutton & Barto, 2020) – action-value and policy-gradient methods. The modularity criterion unlocks the use graphical language for our analysis, which simplifies the proofs. We define a common model of execution for all algorithms within the SDM framework from §3.2 that enables us to compare the causal structures of their different credit assignment mechanisms under ACML. We find that in the general function approximation setting, assuming acyclic decision sequences, the cloned Vickrey society (CVS, §3.2) is the only algorithm to our knowledge so far that produces reinforcement learners that exhibit dynamic modularity.

6.1. From Monolithic Policies to Decision Mechanisms

As mentioned in §3.2, and as motivated by our taqueria example, we are interested in analyzing the independence of different decisions, so we need to adapt the model of execution we gave as an example for MDPs in §4.2 to treat the functions that control each decision as separate mechanisms.

We observe from the SDM framework that any discrete-action policy π with N actions can be decomposed into a set of mechanisms computing a "bid" $b^k_{s_t}$ for each **decision** k (i.e., a value of the action variable, recall §3.2) at the given state s_t, and an independent selection mechanism that selects a decision given the bids (Fig. 3a-c). Define a **decision mechanism** as the function that computes a bid. For policy-gradient methods, a bid corresponds to the action probability for a particular action $p(a = k|\cdot)$, and the selection mechanism is the stochastic sampler for a categorical variable. For action-value methods, a bid corresponds to the estimated Q-value for a particular action, $Q(\cdot, a = k)$, and the selection mechanism could be an ε-greedy sampler or a Vickrey auction (Chang et al., 2020). Often decision mechanisms share weights (e.g. DQN (Mnih et al., 2015)) and thus are algorithmically dependent, but for some algorithms they do not, as in CVS. Then, by absorbing the

[2]This operation is known in λ-calculus as β-reduction.

transition function T and reward function R into APPLY, the function nodes **f** of our model of execution are the decision mechanisms, which each take as input s_t, and produce as output the tuple $(b^k_{s_t}, s_{t+1}, r_t, w^k_t)$, where w_t is a binary flag that indicates whether the selection chose its corresponding action. The execution trace **x**, which we call a **decision sequence**, records the values of these variables in a rollout.

6.2. The Modularity of RL Algorithms

We now ask which action-value and policy-gradient methods exhibit dynamic modularity by evaluating whether their credit assignment mechanisms satisfy the modularity criterion and whether their decision mechanisms share weights.

Which RL algorithms satisfy the modularity criterion? The modularity criterion can be violated if there exists a shared hidden variable in the causal structure of Π that couples together the gradients δ, which causes the δ^k_t's to not be d-separated given **x** and **f** (Fig. 3c-e).

For all policy gradient methods, the gradient into the action probabilities includes a normalization term $\sum_k b^k$ as a shared hidden variable (Fig. 3e):

Corollary 5.1 (policy gradient). *All policy gradient methods do not satisfy the modularity criterion.*

We divide action-value methods into single-step and n-step (where $n > 1$) temporal difference methods, abbrv. TD(0) and TD($n > 1$) respectively. For TD($n > 1$) methods, such as those that use Monte Carlo (MC) estimation of returns, TD(λ) (Sutton, 1985), or generalized advantage estimation (Schulman et al., 2015), this shared hidden variable is a sum of estimated returns or advantages at different steps of the decision sequence (Fig. 3d):

Corollary 5.2 (n-step TD). *All TD($n > 1$) methods do not satisfy the modularity criterion.*

This leaves only TD(0) methods. If the decision mechanism $\mathtt{f}^k$ were selected (i.e. $w^k_t = 1$) at step i, these methods produce, for some function g, gradients as $\delta^k_t := g(b^k_{s_t}, s_t, s_{t+1}, r_t, \mathbf{f})$. Otherwise, $\delta^k_t := 0$. For example, for Q-learning, g is the TD error $[\max_j b^j_{s_{t+1}} + r_t - b^k_{s_t}]$ (Fig. 3f), where $[\max_j b^j_{s_{t+1}}]$ is computed from s_{t+1} and **f**. The only hidden variable is $[\max_j b^j_{s_{t+1}}]$. It is only shared when the decision sequence **x** contains a cycle where two states s_t and s'_t transition into the same state s_{t+1}. In this cyclic case, the credit assignment mechanism would not satisfy the modularity criterion. Otherwise it does:

Corollary 5.3 (single-step TD). *TD(0) methods satisfy the modularity criterion for acyclic* ***x****.*

As cyclic **x** are non-generic cases that arise from specific settings of **x**, we henceforth restrict our analysis to the acyclic case, justifying this restriction similarly to the justification of assuming faithfulness in other causal literature.

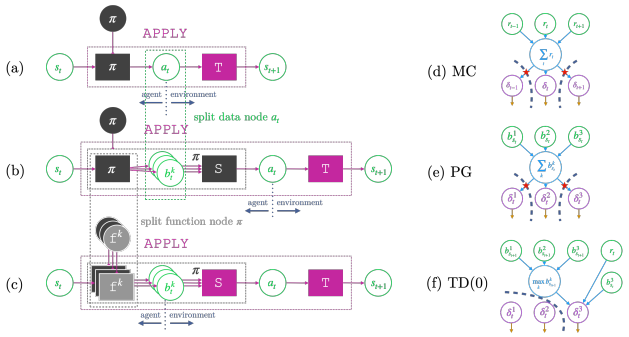

Figure 3: **Modularity in RL.** To convert the MDP model of execution (a) to the SDM model of execution (c), we split the action node into a set of nodes each representing a possible decision (b) and split the monolithic policy into a set of decision mechanisms for each decision (c). The agent-environment boundary separates the learnable decision mechanisms from other algorithmically independent functions, such as the transition function of the MDP (T) or the selection mechanism of the policy (S). TD($n > 1$) methods, like using Monte Carlo (MC) estimation (d), and policy gradient (PG) methods (e) do not have modular credit assignment mechanisms because they contain shared hidden variables. TD(0) methods (f) have modular credit assignment mechanisms in general. The causal edges of non-modular credit assignment cut a partitioning among the gradients δ, indicated by the red star.

Which RL algorithms exhibit dynamic modularity? We have identified TD(0) methods as the class of RL algorithms that satisfy the modularity criterion. By Thm. 3, whether they satisfy dynamic modularity now depends on whether they satisfied static modularity at initialization ($i = 0$). We assume random initialization of **f**, so the only source of dependence among **f** is if they share parameters.

In the tabular setting, decision mechanisms are columns of the Q-table corresponding to each action. Because these columns do not share parameters, Q-learning (Watkins & Dayan, 1992), SARSA (Rummery & Niranjan, 1994), and CVS exhibit dynamic modularity:

Corollary 2.1 (tabular). *In the tabular setting, Thm. 3 holds for Q-learning, SARSA, and CVS.*

In the general function approximation setting, static modularity requires decision mechanisms to not share weights, which eliminates DQN (Mnih et al., 2015) and its variants.

Corollary 2.2 (function approximation). *In the function approximation setting, Thm. 3 holds for TD(0) methods whose decision mechanisms do not share parameters.*

To our knowledge, CVS is the only proposed TD(0) method with this property, but it is straightforward to make existing TD(0) methods exhibit dynamic modularity by using separate networks for estimating the Q-value of each decision.

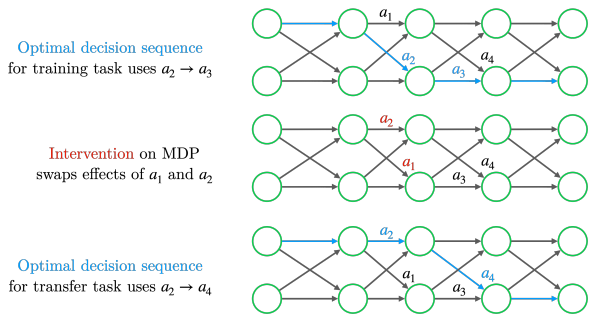

Figure 4: **How transfer tasks are generated.** We consider transfer problems where the optimal decision sequence of the transfer task differs from that of the training task by a single decision. As above, the transfer MDP and the training MDP differ in that the effects of actions a_1 and a_2 get swapped; all other transitions remain the same. The agent must learn to choose action a_4 instead of a_3 while re-using other previously optimal decisions.

Summary. If we want dynamic modularity, then we need the decision mechanisms to not share parameters and the credit assignment mechanism to not contain a shared hidden variable that induced algorithmic dependence among the gradients it outputs. An RL algorithm with dynamic modularity makes it possible for individual decision mechanisms to be modified independently without an accompanying modification to other decision mechanisms.

7. Simple Experiments

This paper is motivated by the hypothesis that modularity enables flexible adaptation. To test this hypothesis requires (1) a method for determining whether a learning algorithm is modular and (2) a metric for evaluating flexible adaptation. The previous sections have contributed (1). The metric we use for (2) is the comparative transfer efficiency of an algorithm that exhibits dynamic modularity with respect to one that does not. We consider transfer problems that require modifying only one decision in a previously optimal decision sequence needs to be changed, similar to our motivating example with Gusteau's taqueria (§1).

Our evaluation focuses on discrete-action on-policy RL algorithms since many factors that influence the learning of off-policy methods are still not well understood (Achiam et al., 2019; Kumar et al., 2020; Van Hasselt et al., 2018; Fu et al., 2019). Specifically we compare three algorithms that span the spectrum of action-value and policy-gradient methods. CVS represents a method that exhibits dynamic modularity. PPO (Schulman et al., 2017) represents a method that is not modular at all. PPOF is a modification of PPO whose where each action logit is computed by a different network, and represents a method that exhibits static modularity at initialization but not dynamic modularity during learning.

We designed our experiments to be as minimal as possible to remove confounders. States are represented as binary vectors. The reward is given at the end of the episode and is 1 if the task is solved and 0 otherwise. The relationship between the training and transfer MDP is given by an intervention in the MDP transition function (Fig. 4).

7.1. An Enumeration of Transfer Problems

Similar to how analysis of d-separation is conducted with triplets of nodes, we enumerated all possible topologies of triplets of decisions: *linear chain*, *common ancestor*, and *common descendant* (Fig. 5, left column). For each topology we enumerated all ways of making an isolated change to an optimal decision sequence. 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 one decision sequence of length three. For example, in Fig. 5, the optimal decision sequence for the *linear chain* training task is $A \to B \to C$. For each topology we have a training task and three independent transfer tasks. Each transfer task represents a different way to modify the MDP of the training task. This single comprehensive task suite (Fig. 5) enables us to ask a wide range of questions. The answers to the questions that follow are scoped only to our stated experimental setup.

Does dynamic modularity improve transfer efficiency? Yes, at least in these experiments. For each of the nine transfer settings (rightmost three columns) in Fig. 5, CVS (red) transfers consistently more efficiently than both PPO (green) and PPOF (blue), despite having comparable training efficiency in the training task (second column from left). The variance among the different runs is also lower for CVS.

How does where a decision needs to be modified in the decision sequence affect transfer efficiency? The improvement in transfer efficiency is especially pronounced in the trend shown in the bottom row of Fig. 5 for *linear chain*. The later the decision that needs to be modified appears in the decision sequence, the wider the gap between CVS and the other two methods, to the point that we had to widen the plot width. Our theory (Thm. 3) offers one possible explanation. Considering the bottom-right plot of Fig. 5, the transfer task requires modifying the last decision and keeping the previous two the same. But the lack of independent gradients and parameters in PPO and PPOF seems to have affected correct decision mechanisms in the first two steps based on the errors encountered by the decision mechanism in the last step, seemingly causing the previous decision mechanisms to "unlearn" originally optimal behavior, then relearn the correct behavior again, as shown in the plots for "state 0" and "state 1" in Fig. 6. This unnecessary unlearning and relearning seems to be a primary reason for the lower transfer efficiency of PPO and PPOF. It is as if Colette in

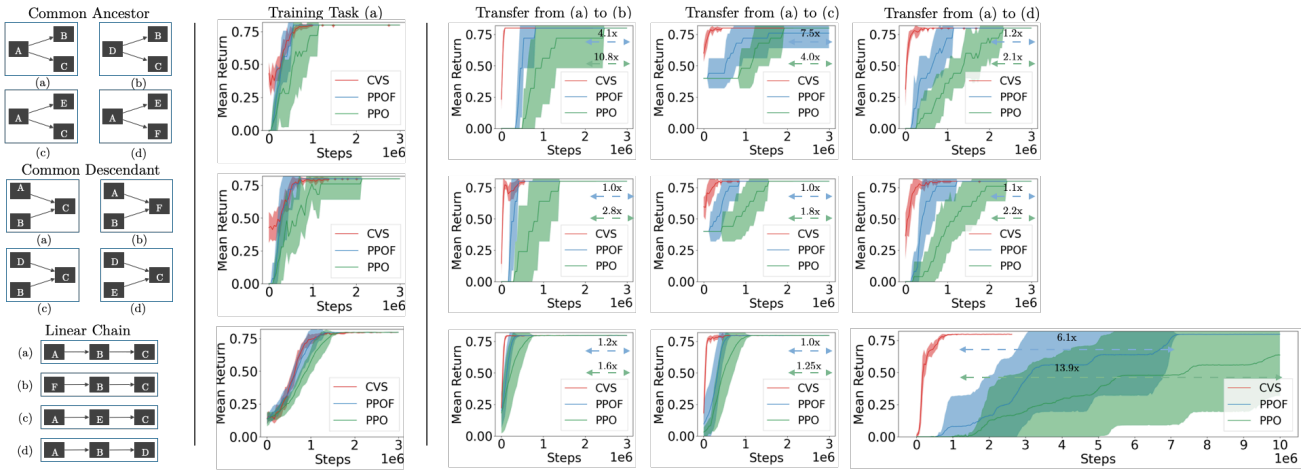

Figure 5: **Transfer problems involving triplets of decisions.** For each task topology (leftmost column) we have a training task, labeled (a) and three independent transfer tasks, labeled (b,c,d). Each transfer task is a different way to modify the training task's MDP. CVS consistently exhibits higher sample efficiency than both PPO and PPOF showing that dynamic modularity correlates with more efficient transfer. Notably the gap between CVS and the other methods in the bottom-right (e.g. 13.9x more efficient than PPO) is so wide that we had to extend the chart width. 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. Shown are runs across ten seeds.

Gusteau's taqueria (§1) stopped heating tortillas because of the angry reviews about meat contamination but then realized that she should still be heating tortillas after all.

Does dynamic modularity enable independent modification of decision mechanisms in practice? While theory tells us that decision mechanisms can be modified independently within a single credit assignment update, in practice transfer learning requires multiple credit assignment updates to converge. Across multiple credit assignment updates, the decision mechanisms would no longer be independent, even for algorithms that exhibit dynamic modularity, but it is also expected that the functions of a learner should learn to work together over the course of learning in any case. Nonetheless, Fig. 6 shows that the lack of a softmax tying the bids of CVS together enables them to change more independently and rapidly than PPOF.

Figure 6: **How the decision mechanisms change during transfer.** Shown the three states of the decision sequence. The optimal last decision must change from action C (purple) to action D (green). CVS modifies its bids independently. The bids for PPOF are coupled together across decision mechanisms and across time.

How much of transfer efficiency is due to modular credit assignment than network factorization? This question pits our theory against a competing explanation: that network factorization alone (represented by PPOF) is responsible for improved transfer efficiency. Though PPOF is more efficient than PPO in training and transfer, PPOF is consistently less efficient than CVS in transfer while being similarly efficient in training. This suggests that network factorization is not a sufficient explanation, leaving our theory of dynamic modularity still standing.

7.2. Modularity and Forgetting

A desirable consequence of having the capacity to independently modify learnable mechanisms is the ability to *not* modify mechanisms that need not be modified: we would not want the agent to forget optimal behavior in one context when it trains on a different task in a different context. We now test whether dynamic modularity contributes to this ability. The experimental setup is shown in Fig. 7. There are four possible values for the action, A, B, C, D. In task (a), the optimal decision sequence is $A \rightarrow C$, starting at state s_0 and passing through state s_2, which has a context bit flipped to 0. In task (b), the optimal decision sequence is $B \rightarrow D$, starting at state s_1 and passing through state s_2, which has a context bit flipped to 1. Though the optimal states for task (a) are disjoint from the optimal states for task (b), the decision mechanisms corresponding to A, B, C, D are present for both tasks. We first train on task (a), then transfer from (a) to (b), then transfer back from (b) to (a).

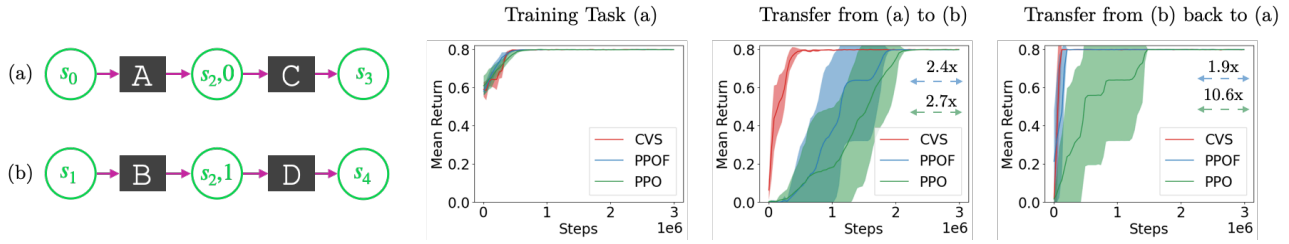

Figure 7: **Modularity and forgetting.** The optimal solutions for tasks (a) and (b) involve a disjoint set of decisions: $A \rightarrow C$ for task (a) and $B \rightarrow D$ for task (b). We first train on task (a), then transfer from (a) to (b), then transfer back from (b) to (a). The purpose of this experiment is to test whether dynamic modularity improve the agent's ability to preserve optimal behavior on a previous task after having trained to convergence on a different task in a different context. While both CVS and PPO have similar sample efficiency when initially training on task (a), CVS is more than ten times more sample efficient than PPO when transferring back from (b) to (a), suggesting that PPO "forgot" the optimal behavior for task (a) when training on task (b), which is not the kind of forgetting we want in learning agents.

Does dynamic modularity improve the agent's ability to preserve optimal behavior on a previous task after having trained to convergence on a different task? To test this, we compare CVS and PPO's sample efficiency when transferring back from (b) to (a). Fig. 7 shows that even when both CVS and PPO have similar sample efficiency when initially training on task (a), CVS is more than ten times more sample efficient than PPO when transferring back from (b) to (a). Our explanation for this phenomenon is that the lack of algorithmic independence in the decision mechanisms of PPO causes the decision mechanisms for actions A and C to be significantly modified when PPO transfers from (a) to (b), even when these actions do not even participate in the optimal decision sequence for task (b). The low sample efficiency when transferring back from (b) to (a) suggests that PPO "forgot" the optimal behavior for task (a) when training on task (b), which is not the kind of forgetting we want in flexibly adaptable agents.

How much of this ability to preserve previously optimal behavior due to modular credit assignment than network factorization? PPOF is similarly inefficient as PPO compared to CVS in transferring from (a) to (b), which is consistent with our findings from §7.1. Interestingly, PPOF seems to be just as efficient at transferring back from (b) to (a) as CVS, which seems to suggest that the primary cause for the forgetfulness of PPO, at least in this experiment, is less due to lack of independent gradients but more to lack of network factorization. This experiment suggests a need for an explanatory theory to identify under which circumstances independent gradients are more influential to flexible adaptation than network factorization, and vice versa, as well as a means for quantifying the degree of influence each has.

8. Discussion

The hypothesis that modularity can enable flexible adaptation requires a method for determining whether a learning system is modular. This paper has contributed the modularity criterion (Thm. 5) as such a method. The consistency of how dynamic modularity in on-policy reinforcement learning correlates with higher transfer efficiency in our experiments suggests a need for future work to provide an explanatory theory for exactly how dynamic modularity contributes to flexible adaptation as well as to test whether the same phenomenon can be observed with other classes of learning algorithms, other transfer problems, and other domains. The modularity criterion is a binary criterion on algorithmic independence or lack thereof, but our experiments also suggest a need for future work in quantifying algorithmic causal influence if we want to relax the criterion to a softer penalty on algorithmic mutual information. Learning algorithms are simply only one example of the more general concept of the dynamic computational graph introduced in this paper, which we have shown with Lemma 4 can be used to analyze the algorithmic independence of functions that evolve over time. The connection we have established among credit assignment, modularity, and algorithmic information theory, in particular the link between learning algorithms and algorithmic causality, opens many opportunities for future work, such as new ways of formalizing inductive bias in the algorithmic causal structure of learning systems *and* the learning algorithms that modify them.

Acknowledgements

We thank Daniel Filan, Michael Janner, Giambattista Parascandolo, Sam Toyer, and Olivia Watkins for feedback on earlier drafts of the paper. We thank Dominik Janzing, Aviral Kumar, and Jason Peng for useful discussions. This research was supported by the AFOSR grant FA9550-18-1-0077, and computing resources from Amazon Web Services and Microsoft Azure. MC is supported by the National Science Foundation Graduate Research Fellowship Program.

References

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

Achiam, J., Knight, E., and Abbeel, P. Towards characterizing divergence in deep q-learning. *arXiv preprint arXiv:1903.08894*, 2019.

Aldrich, J. Autonomy. *Oxford Economic Papers*, 41(1): 15–34, 1989.

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

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

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

Balduzzi, D., Czarnecki, W. M., Anthony, T. W., Gemp, I. M., Hughes, E., Leibo, J. Z., Piliouras, G., and Graepel, T. Smooth markets: A basic mechanism for organizing gradient-based learners. *arXiv preprint arXiv:2001.04678*, 2020.

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

Bird, B., Lewis, B., Pinkava, J., and Capobianco, J. Ratatouille, 2007.

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

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

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

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

Csordás, R., van Steenkiste, S., and Schmidhuber, J. Are neural nets modular? inspecting functional modularity through differentiable weight masks. *arXiv preprint arXiv:2010.02066*, 2020.

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

Filan, D., Casper, S., Hod, S., Wild, C., Critch, A., and Russell, S. Clusterability in neural networks. *arXiv preprint arXiv:2103.03386*, 2021.

Ford, M. *Architects of Intelligence: The truth about AI from the people building it*. Packt Publishing Ltd, 2018.

Fu, J., Kumar, A., Soh, M., and Levine, S. Diagnosing bottlenecks in deep q-learning algorithms. In *International Conference on Machine Learning*, pp. 2021–2030. PMLR, 2019.

Gács, P., Tromp, J. T., and Vitányi, P. M. Algorithmic statistics. *IEEE Transactions on Information Theory*, 47 (6):2443–2463, 2001.

Gemp, I., McKee, K. R., Everett, R., Duéñez-Guzmán, E. A., Bachrach, Y., Balduzzi, D., and Tacchetti, A. D3c: Reducing the price of anarchy in multi-agent learning. *arXiv preprint arXiv:2010.00575*, 2020.

Goodman, N. D., Tenenbaum, J. B., and Contributors, T. P. Probabilistic Models of Cognition. http://probmods.org/v2, 2016. Accessed: 2021-1-29.

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

Haavelmo, T. The probability approach in econometrics. *Econometrica: Journal of the Econometric Society*, pp. iii–115, 1944.

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

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

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

Kumar, A., Agarwal, R., Ghosh, D., and Levine, S. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020.

Lemeire, J. Conditional independencies under the algorithmic independence of conditionals. *The Journal of Machine Learning Research*, 17(1):5252–5271, 2016.

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

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

Mansinghka, V. K. et al. *Natively probabilistic computation*. PhD thesis, Massachusetts Institute of Technology, Department of Brain and Cognitive . . . , 2009.

Meek, C. Strong completeness and faithfulness in bayesian networks. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, pp. 411–418, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1558603859.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

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

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

Pearl, J. *Causality*. Cambridge university press, 2009.

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

Rummery, G. A. and Niranjan, M. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

Samejima, K., Doya, K., and Kawato, M. Inter-module credit assignment in modular reinforcement learning. *Neural Networks*, 16(7):985–994, 2003.

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

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

Simpkins, C. and Isbell, C. Composable modular reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4975–4982, 2019.

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

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

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

Sprague, N. and Ballard, D. Multiple-goal reinforcement learning with modular sarsa (0). 2003.

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

Sutton, R. S. Temporal credit assignment in reinforcement learning. 1985.

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

van de Meent, J.-W., Paige, B., Yang, H., and Wood, F. An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*, 2018.

Van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

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

Watanabe, C. Interpreting layered neural networks via hierarchical modular representation. In *International Conference on Neural Information Processing*, pp. 376–388. Springer, 2019.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.