Reinforcement Learning in Categorical Cybernetics

Jules Hedges   Riu Rodríguez Sakamoto

We show that several major algorithms of reinforcement learning (RL) fit into the framework of categorical cybernetics, that is to say, parametrised bidirectional processes. We build on our previous work in which we show that value iteration can be represented by precomposition with a certain optic. The outline of the main construction in this paper is: (1) We extend the Bellman operators to parametrised optics that apply to action-value functions and depend on a sample. (2) We apply a representable contravariant functor, obtaining a parametrised function that applies the Bellman iteration. (3) This parametrised function becomes the backward pass of another parametrised optic that represents the model, which interacts with an environment via an agent. Thus, parametrised optics appear in two different ways in our construction, with one becoming part of the other. As we show, many of the major classes of algorithms in RL can be seen as different extremal cases of this general setup: dynamic programming, Monte Carlo methods, temporal difference learning, and deep RL. We see this as strong evidence that this approach is a natural one and believe that it will be a fruitful way to think about RL in the future.

1   Introduction

Reinforcement learning (RL) refers to a class of methods in machine learning for optimising a long-run reward during interaction with an unknown environment. It is considered one of the major pillars of machine learning, along with *deep learning* (neural networks and differentiable programming), *unsupervised learning* (statistical clustering methods, which includes topological data analysis [Ghr08]) and *variational learning* (Bayesian inference and related probabilistic methods). It can be seen as an extension of *dynamic programming* methods in optimal control theory [Ber19], which drops the assumption that a model of the environment is known. RL, combined with deep learning methods to produce *deep RL*, notably achieved state of the art success in practical game playing, with AlphaGo [SSS+17] defeating the human Go champion in 2016 and AlphaStar [VBC+19] achieving Grandmaster status in the real time strategy game StarCraft II.

In this paper we show that several major algorithms of reinforcement learning fit into the framework of *categorical cybernetics*, that is to say, *parametrised bidirectional processes* [CGHR22]. This branch of applied category theory has already been applied to deep learning [CGG+22, Gav24], variational learning [BHS23, Smi23] and game theory [GHWZ18, BHZ23]. It is also a close relative of the *categorical systems theory* of Myers, Spivak and others [Mye23, NS24].

We build on our previous work [HS23] in which we show that *value iteration*, a fundamental method common to both dynamic programming and RL, can be represented (in the technical sense) by precomposition with a certain optic. Specifically, for each *policy* π we define an *optic* $\mathbb{B}(\pi) : \binom{S}{\mathbb{R}} \to \binom{S}{\mathbb{R}}$, where S is the set of states of a Markov decision process. This has the property that for any value function *value function* $V : S \to \mathbb{R}$, represented as an optic $V : \binom{S}{\mathbb{R}} \to I$, $V \circ \mathbb{B}(\pi)$ is a better value function. This precomposition with $\mathbb{B}(\pi)$ is called a *Bellman operator*.

The outline of the main construction in this paper is: (1) We extend $\mathbb{B}$ to a *parametrised optic* representing a more general class of Bellman operators that apply to *action-value functions* and depend

DRAFT   April 4, 2024

on a *sample* as a parameter. (2) We apply $\mathbb{K}$, a representable contravariant functor that already plays a foundational role in compositional game theory, obtaining a parametrised function $\mathfrak{B} = \mathbb{K}(\mathbb{B})$ that applies the Bellman iteration. (3) This parametrised function becomes the backward pass of another parametrised optic that represents the *model*, which interacts with an *environment* via an *agent*. Thus, parametrised optics appear in two different ways in our construction, with one becoming part of the other. This stays within the existing ingredients of categorical cybernetics, but combines them in a way that has not been seen elsewhere.

As we show, many of the major classes of algorithms in RL can be seen as extremal cases of this general setup: dynamic programming, Monte Carlo methods, temporal difference learning, and deep RL. We see this as strong evidence that this approach is a natural one and believe that it will be a fruitful way to think about RL in the future. Although we focus on single-agent RL, the compositionality of our methods makes them naturally well-suited to *multi-agent* RL, which is a close relative of game theory.

2   Background: Reinforcement learning

Algorithms in RL specify how agents learn optimal behaviours through interaction with their environment. This interaction provides feedback to actions, and is the key feature that differentiates it with respect to supervised and unsupervised learning. The fundamental goal of RL is to enable agents to make sequential decisions in dynamic environments to maximize long-term cumulative rewards. This process involves the agent taking actions, observing the resulting states and rewards, and using this information to update its decision-making strategy over time.

Our approach to study these algorithms is structural, and the main structural distinction is between the agent and the environment. The **environment** represents the external system with which the agent interacts, and is assumed be a Markov decision process. To quickly recall, a Markov process (MP) consists of a set of states S and a stochastic transition function $t : S \rightarrow DS$, where D is some probability monad over **Set**. A Markov reward process (MRP) is a Markov process with an additional *utility* function that outputs rewards $u : S \rightarrow D\mathbb{R}$. A Markov decision process (MDP) is a MRP with a set A of actions, whose transition and utility functions use the action taken at each state, as in $\langle t, u \rangle : S \times A \rightarrow D(S) \times D(\mathbb{R})$. An agent's goal is to maximize the *expected long-run reward* $\sum \gamma^i r_i$, where $0 < \gamma \leq 1$ is a hyperparameter called the **discount factor** which controls the agent's "patience", or preference between rewards in the present and rewards in the future.

The environment's response to an agent's action is given by transition dynamics that can be assumed to have the Markov property, and the environment's state is known to the agent. When the agent only has access to a partial observation of the state, we speak of a partially observable MDP (POMDP).

The **agent** has as core components the policy, the reward, the value function and the internal model. A **policy** $\pi : S \rightarrow TA$ or scheduler defines its strategy mapping states to actions. The policy is either single-valued or deterministic ($T = 1$ the identity functor), many-valued ($T = \mathcal{P}$ the powerset functor, like argmax) or probabilistic ($T = D$ the distribution functor, like ε-greedy), with probabilistic being the most common. The **reward** is the immediate response of the environment after an action, and the maximization of its expected cumulative sum is the goal of the agent under the *reward hypothesis* [SSPS21]. A **value function** estimates this expected long-term reward associated with following a particular policy. Usually one works with either a state value function $V : S \rightarrow \mathbb{R}$ or a state-action value function $Q : S \times A \rightarrow \mathbb{R}$, where $V(s)$ estimates the long-run reward of following a certain policy from each state, and $Q(s, a)$ estimates the long-run reward of taking each action in each state and then following a certain policy after that. Both policies and value functions are characterised as solutions of functional equations

known as **Bellman equations**, using the temporal 'self-similarity' of MDPs.

When S and A are finite sets the function Q is typically implemented as a mutable lookup table called a Q-table or Q-matrix. A **model** is an approximation or representation of the environment's dynamics, allowing the agent to simulate or predict future states and rewards. One surrogate objective of an agent is to improve its model. Not all agents have models, so there's a distinction between **model-based** and **model-free** methods.

Methods whose policies for environment interaction π_{beh} ("behaviour policy") are different to the ones for model improvement π_{tgt} ("target policy") are called **off-policy**, while **on-policy** methods only have one policy. Finally, another distinction is drawn between **online** and **offline** or **batch RL** methods, where the former family learns while interacting with the environment, while the latter learns from pre-recorded experiences.

RL encompasses many algorithms and methodologies, including dynamic programming, Monte Carlo methods, temporal difference learning, deep reinforcement learning, and more. This diversity of methods employs experimental and formal justifications to tackle weak spots in this learning theory such as the credit-assignment problem, the exploration-exploitation tradeoff and coping with state that is hidden or too big to represent explicitly. Many of these were problems already identified in preceding fields such as psychology and neuroscience [KLM96].

2.1 Dynamic programming

Dynamic programming (DP) methods are an idealized class of model-based algorithms that do not need to interact with the environment because they have a perfect model of it as an MDP. They are not usually used in their classical formulation that we describe next in practical settings because of the perfect model assumption and their high computational expense, and serve rather as a theoretical baseline to approximation methods and other RL techniques.

The idea behind DP is to treat the Bellman equation for the optimal value of a policy and the Bellman equation for the optimal policy of a value function as update operators on a space of value functions.

The search for an optimal policy happens entirely within the agent's model, interleaving two feedback operations called the *value improvement* and *policy improvement* steps which treat the Bellman equations as update rules. This process of updating previous estimates is called **bootstrapping**.

- **Value improvement** or policy evaluation updates the value function $V : S \to \mathbb{R}$ pointwise by traversing the state space S and updating the state's estimated value $V(s)$ with the expected discounted value after one simulation step:

$$V(s) \leftarrow \mathbb{E}_{\substack{a \sim \pi(s) \\ (s',r) \sim t(s,a)}} [r + \gamma V(s')] = \sum_{a \in A} \pi(s,a) \sum_{\substack{s' \in S \\ r \in \mathbb{R}}} t(s,a,s',r)(r + \gamma V(s')) \tag{1}$$

  Here we are interchangeably considering a stochastic function $f : X \to DY$ as a function $f : X \times Y \to [0,1]$. The sum over $\mathbb{R}$ makes sense when D is finite support probability distributions, and in more general settings is replaced with an integral. We write $\mathfrak{B}_{\text{val}}(V,\pi)(s) = \mathbb{E}[r + \gamma V(s')]$ for the operator $\mathfrak{B}_{\text{val}} : \mathbb{R}^S \times A^S \to \mathbb{R}^S$. We will discuss Bellman operators in Section 5.

- **Policy improvement** updates the policy function $\pi : S \to A$ pointwise by traversing the state space S and updating the action taken in the state $\pi(s)$ with $\text{argmax}_a \mathbb{E}_{(s',r) \sim t(s,a)} [r + \gamma V(s')]$. Similarly, we write $\mathfrak{B}_{\text{pol}}(V)(s) = \text{argmax}_a \mathbb{E}_{s',r \sim t(s,a)} [r + \gamma V(s')]$ for the operator $\mathfrak{B}_{\text{pol}} : \mathbb{R}^S \to A^S$.

Depending on the sequencing of these two steps, we have three classic algorithms, where we write $(-)^\dagger$ for the (in practice approximate) fixpoint of the operator and $\overline{\mathfrak{B}_{\text{pol}}}(V,\pi) = (V, \mathfrak{B}_{\text{pol}}(V))$ and $\overline{\mathfrak{B}_{\text{val}}}(V,\pi) =$

$(\mathfrak{B}_{\mathrm{val}}(V,\pi),\pi)$ for the embeddings of the two Bellman operators as maps $\mathbb{R}^S \times A^S \to \mathbb{R}^S \times A^S$: **policy iteration** (PIT) as $(\overline{\mathfrak{B}_{\mathrm{pol}} \circ \overline{\mathfrak{B}_{\mathrm{val}}}^\dagger})^\dagger$, **value iteration** (VIT) as $(\overline{\mathfrak{B}_{\mathrm{pol}} \circ \overline{\mathfrak{B}_{\mathrm{val}}}})^\dagger$ and **generalized policy iteration** (GPI) as $(\overline{\mathfrak{B}_{\mathrm{pol}}}^m \circ \overline{\mathfrak{B}_{\mathrm{val}}}^n)^\dagger$ for $m, n > 0$.

2.2 Monte Carlo

Monte Carlo (MC) methods are antithetical to DP, because they don't assume any prior knowledge of the environment's dynamics. Without this knowledge, the way to learn the value function and obtain a optimal policy is to estimate it from sample trajectories. Averaging over many trajectories should converge to the expected value.

The agent's internal model consists of a value function $Q : S \times A \to \mathbb{R}$, from which a policy like the ε-greedy $\pi : S \to DA$ is derived: $\pi(s) = \mathrm{argmax}_a Q(s,a)$ with probability $1 - \varepsilon$ and uniformly random between all actions with probability ε. The value function improvement is pointwise, but unlike DP, MC improves $Q(s,a)$ by averaging over many returns that start at (s,a). Given a single episode $(s,a,r,s',a',r',\dots)$ starting at (s,a), the update **target** becomes $G = \sum_t \gamma^t r_t$, and the value function updates as

$$Q(s,a) = (1 - \alpha)Q(s,a) + \alpha G \tag{2}$$

where the learning rate $\alpha : [0,1]$ is a step size hyperparameter. Note that the lack of bootstrapping is shown by the fact that G does not contain any reference to the value function.

2.3 Temporal difference learning

Temporal difference learning (TD) is a class of methods that learn from both the interaction with the environment (MC's sampling) and from previous estimates of the value function (DP's bootstrapping).

Given a finite episode $(s,a,r,\dots,s_n,a_n)$ starting at (s,a), we can modify the target for (2) to consist of the discounted sum of the $n-1$ returns and an estimated long-run value of the last state-action pair. We write n-TD for the class of TD methods whose trajectories contain n return values.

Example 2.1 (SARSA [Sut95])**.** SARSA is a 1-TD on-policy control method, which updates the (s,a)-indexed Q-value with the target $G = r + \gamma Q(s',a')$. The name originates from the model feedback consisting of a 1-step episode (s,a,r,s',a'). Some variants of SARSA include n-SARSA, with $G = \sum_{t=1}^{n-1} \gamma^t r_t + \gamma^n Q(s_n,a_n)$, and Exp-SARSA, with $G = r + \gamma \mathbb{E}_{a \sim \pi_{\mathrm{tgt}}(s)} Q(s,a)$, which is off-policy because the last action is determined in expectation by a target policy π_{tgt}.

Example 2.2 (Q-learning [WD92])**.** In Q-learning, given the current state s the agent performs an action $a \sim \pi_{\mathrm{beh}}(s)$ using a policy derived from its internal Q-table, for example an ε-greedy policy, and gets from the environment the reward r and the next state s'. The feedback to the model is the tuple (s,a,r,s'). The model then updates its Q-table with its target policy $G = r + \gamma Q(s, \pi_{\mathrm{tgt}}(s)) = r + \gamma \max_{a' \in A} Q(s',a')$. It is an off-policy method because the last action used to compute the update is $\pi_{\mathrm{tgt}}(s') = \mathrm{argmax}_{a' \in A} Q(s',a')$ and not $\pi_{\mathrm{beh}}(s')$.

Q-learning is the first appearance of a major subtlety of RL: the distinction between actions that the agent actually performs during an interaction with its environment, and actions which are "internal" or "simulated". The actions that the agent actually performs in Q-learning are always drawn from the policy π_{beh}, whereas the action $\mathrm{argmax}_{a' \in A} Q(s',a')$ is used only when computing updates. We can consider this to be a separate target policy, $\pi_{\mathrm{tgt}}(s') = \mathrm{argmax}_{a' \in A} Q(s',a')$.

2.4 Approximation methods

The methods seen so far are usually denoted *tabular* methods, because they work around an encoding of value and/or policy functions as lookup tables, taking into account that the state and action spaces S, A are finite sets of manageable cardinality. Approximation methods tackle the state space explosion problems that arise when these sets become too big, e.g. the *curse of dimensionality* in continuous state space settings or even in situations where the observation of an image as a pixel space begs for lossy encodings. Among splines, hierarchical methods, memory-based methods, linear methods with (kernel) features and others, we focus on supervised-learning function approximation methods or **gradient-based** methods, which are known as *deep reinforcement learning*.

Deep Q-networks (DQN) [MKS$^+$13] are a notable method in this area. One can motivate them by observing the Q-table update $Q, (s, a, r, s') \mapsto Q'$ of Q-learning as a sampled discrete (semi)gradient[1] update. Let $\mathcal{L} = (Q(s, a) - G)^2$ be a loss function, which quantifies the discrepancy between the Q-value and the sample target. The update equation (2) becomes $Q = Q - \frac{\alpha}{2} \partial_{Q(s,a)} \mathcal{L}$. A DQN parametrises the Q-table as a function $\Theta \times S \times A \to \mathbb{R}$. For performance considerations, the function is curried into $\mathrm{DQN} : \Theta \times S \to (A \to \mathbb{R})$, which is implemented as a neural network with $|S|$ inputs and $|A|$ outputs with values in $\mathbb{R}$. The encoding of the Q-learning method as a neural network leverages all of the powerful techniques of deep learning.

In DQN, the policy is still obtained as a function of the now approximate Q-function. **Policy gradient** (PG) methods [SMSM99] bypass the generation of a value altogether, by directly outputting an action distribution $\mathrm{PG} : \Theta \times S \to DA$. One can see DQNs as a special PG method by transforming the Q-function into an action probability via the Boltzmann distribution, also known as softmax.

3 Background: categorical cybernetics

In this section we quickly recall the main ideas of categorical cybernetics, mostly from [CGHR22].

3.1 Actegories

Given a monoidal category $\mathcal{M}$ and a category C, an action of $\mathcal{M}$ on C, also called an **actegory**, is a functor $\bullet : \mathcal{M} \times C \to C$ together with coherent isomorphisms $I \bullet X \cong X$ and $(M \otimes N) \bullet X \cong M \bullet (N \bullet X)$ [CG23]. Every monoidal category has a self-action given by $\otimes : C \times C \to C$.

If $\mathcal{M}$ and C are monoidal categories and $F : \mathcal{M} \to C$ is a strong monoidal functor, then $M \bullet X = F(M) \otimes X$ is an actegory. A coherent action of one symmetric monoidal category on another, called a **symmetric actegory**, is necessarily of this form. For example, the self-action of a symmetric monoidal category is a symmetric actegory given by the identity functor $C \to C$. All actegories in this paper will be symmetric.

An **enrichment** of a category C in a monoidal category $\mathcal{M}$ is a functor $[-, -] : C^{\mathrm{op}} \times C \to \mathcal{M}$ plus additional data and conditions [Kel82]. There is a tight connection between actegories and enrichments: if $\bullet$ is any actegory such that every $- \bullet X : \mathcal{M} \to C$ has a right adjoint $[X, -] : C \to \mathcal{M}$ (called a closed actegory) then $[-, -]$ is an enrichment, and conversely if $[-, -]$ is an enrichment such that every $[X, -]$ has a left adjoint $- \bullet X$ (called a copowered or tensored enrichment) then $\bullet$ is an action. For example, if C is any category with all coproducts then it has a tensored enrichment in **Set** and therefore an action $\bullet : \mathbf{Set} \times C \to C$ given by $M \bullet X = \sum_M X$.

---

[1] Semi-gradient because the Q-function is contained in the target yet it is considered constant in the gradient computation [SB20, Sec.9.3][Bar93].

3.2   Parametrisation

Given an actegory $\bullet : \mathcal{M} \times C \to C$, a **parametrised morphism** $f : X \to Y$ in C is a pair of an object $M : \mathcal{M}$ and a morphism $f : M \bullet X \to Y$. The identity parametrised morphism is given by $I : \mathcal{M}$ and $\mathrm{id}_X : I \bullet X \cong X \to X$. The composite of $(M, f : M \bullet X \to Y)$ and $(N, g : N \bullet Y \to Z)$ has parameter $N \otimes M$ and morphism $(N \otimes M) \bullet X \xrightarrow{\cong} N \bullet (M \bullet X) \xrightarrow{N \bullet f} N \bullet Y \xrightarrow{g} Z$. A reparametrisation from $(M, f : M \bullet X \to Y)$ to $(N, g : N \bullet X \to Y)$ is a morphism $h : M \to N$ in C such that $f = g \circ (h \bullet X)$.

   Given an actegory $\bullet : \mathcal{M} \times C \to C$, we have a bicategory whose objects are objects of C, 1-cells are parametrised morphisms and 2-cells are reparametrisations. This may be referred to by $\mathbf{Para}_{\mathcal{M}}(C)$, $\mathbf{Para}_{\bullet}(C)$ or simply $\mathbf{Para}(C)$ when unambiguous. We believe that when $\bullet$ is a symmetric monoidal actegory, $\mathbf{Para}_{\bullet}(C)$ is a symmetric monoidal bicategory [HS19] but this has not yet been proven.

   When we have an action $\bullet : \mathcal{M} \times C \to C$ and a symmetric lax monoidal functor $W : \mathcal{M} \to \mathbf{Set}$ (or sometimes $\mathbf{Cat}$), and we extend $\bullet$ to an action of the category of elements $\int W$ by precomposing with the discrete fibration $\pi : \int W \to \mathcal{M}$ to obtain $(M, w) \bullet X = M \bullet X$. When W is lax monoidal with laxator $\nabla : W(X) \times W(Y) \to W(X \otimes Y)$, $\int W$ gains a symmetric monoidal product $(X, w_X) \otimes (Y, w_Y) = (X \otimes Y, w_X \nabla w_Y)$. We write $\mathbf{Para}^W_{\mathcal{M}}(C)$ for $\mathbf{Para}_{\int W}(C)$, and call this **weighted parametrisation** [Gav24].

   Dually, a **coparametrised morphism** $f : X \to Y$ is a pair of an object $M : \mathcal{M}$ and a morphism $f : X \to M \bullet Y$. There is a category $\mathbf{CoPara}_{\bullet}(C)$ of objects, coparametrised morphisms and reparametrisations.

   Given a category C enriched in a monoidal category $\mathcal{M}$, an **externally parametrised morphism** $f : X \to Y$ of C is a pair of an object M of $\mathcal{M}$ and a morphism $f : M \to [X, Y]$ of $\mathcal{M}$ [Smi23]. There is once again a bicategory $\mathbf{Para}_{\mathcal{M}}(C)$ of externally parametrised morphisms. In the case of a tensored enrichment this bicategory is equivalent to the previous one, but there are also interesting cases when they differ. Coparametrised morphisms cannot be defined for an enrichment that is not tensored.

3.3   Optics

Given a monoidal category $\mathcal{M}$ acting on categories C and $\mathcal{D}$, and given objects X, Y of C and X', Y' of $\mathcal{D}$, a **mixed optic** $\binom{X}{X'} \to \binom{Y}{Y'}$ is an equivalence class of triples of an object $M : \mathcal{M}$, a coparametrised morphism $f : X \to M \bullet Y$ of C and a parametrised morphism $f' : M \bullet Y' \to X'$ of $\mathcal{D}$. The equivalence classes are generated by reparametrisations and satisfy the universal property of a coend, $\int^{M : \mathcal{M}} C(M \bullet X, Y) \times \mathcal{D}(M \bullet Y', X')$. There are two different string diagram notations for an optic (figure 1). The first considers them as morphisms of a monoidal category, composing left-to-right, with causality flowing clockwise from top-left. The second considers them as colours of an operad, composing outside-in, with causality flowing left-to-right.
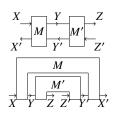
Figure 1: Alternative notations for optic composition

   There is a category $\mathbf{Optic}_{\mathcal{M}}(C, \mathcal{D})$ whose objects are pairs and whose morphisms are optics. When both actions are symmetric, or equivalently are defined by a span of symmetric monoidal functors $C \leftarrow \mathcal{M} \to \mathcal{D}$, then $\mathbf{Optic}_{\mathcal{M}}(C, \mathcal{D})$ is a symmetric monoidal category, with the tensor product on objects being pairwise.

   In the common case that $\mathcal{M} = C = \mathcal{D}$ acts on itself by monoidal product, we write $\mathbf{Optic}(C)$. The tensor product of $\mathbf{Optic}(C)$ is pairwise monoidal product. When the monoidal unit of C is terminal (which includes all Markov categories) then we have natural isomorphisms $\mathbf{Optic}(C)\left(I, \binom{X}{X'}\right) \cong C(I, X)$ and $\mathbf{Optic}(C)\left(\binom{X}{X'}, I\right) \cong C(X, X')$. We call morphisms in latter case **continuations**, and define the representable functor $\mathbb{K} = \mathbf{Optic}(C)(-, I) : \mathbf{Optic}(C)^{\mathrm{op}} \to \mathbf{Set}$.

There are two common cases when the coend in the definition of optics can be eliminated using the ninja Yoneda lemma [Ril18, Lor21]. Firstly, when $\mathcal{M} = C = \mathcal{D}$ acts on itself by cartesian product then there is a natural isomorphism $\mathbf{Optic}\left(\binom{X}{X'}, \binom{Y}{Y'}\right) \cong C(X, Y) \times C(X \times Y', X')$. This is usually known as a **Lens**. Although this case is much easier to understand, there are significant conceptual advantages to the more general definition [Gav22]. Secondly, when $\mathcal{M} = C = \mathcal{D}$ acts on itself by a closed monoidal product then there is a natural isomorphism $\mathbf{Optic}\left(\binom{X}{X'}, \binom{Y}{Y'}\right) \cong C(X, Y \otimes [Y', X'])$. Both of these cases can be generalised to requiring a condition on only one side.

For the cartesian self-action of **Set**, **Optic**(**Set**) coincides with the category of monomial endofunctors and natural transformations. Any cartesian self-action in a category with finite limits can be generalised to **dependent lenses** (also known as morphisms of **containers** [AAG03]), which in the locally cartesian closed case are equivalent to **polynomial endofunctors** [NS24]. Finding the best of both worlds between the monoidal and cartesian cases is known as **dependent optics** [Ver23] and is only partially understood. There are reasons to want to use dependent optics in this paper because it is common that the available actions of a reinforcement learning agent depends on the current state of the Markov chain [BJI+17], but we only consider the simply-typed case in this paper for simplicity.

3.4 Parametrised optics

When a category of optics is symmetric monoidal, it admits a self-action. In [CGHR22] it was identified that the resulting category **Para**(**Optic**) of **parametrised optics** is extremely rich, and provides a general-purpose foundation for the study of controlled processes. The study of this is known as **categorical cybernetics**, which includes compositional game theory [Dal19], deep learning [CGG+22], compositional Bayesian inference [BHS23] and variational learning [Smi23], and applications in software engineering such as *open servers* [VC22].

4 States, contexts and iteration

An optic (whether parametrised or not) is a process consisting of a forward pass followed by a backward pass. In many applications, including those in this paper, this process is iterated through repeated interaction with an outside environment. In the case of supervised learning, this could simply be samples drawn from a dataset. In this section we will develop a general theory of iterated optics.

Let C be a symmetric monoidal category and $W : C \to \mathbf{Set}$ a symmetric lax monoidal functor. Consider the bicategory $\mathbf{Para}^W(C)$ generated by the action of $\int W$ on C given by $(M, w) \bullet X = M \otimes X$. A parametrised morphism $X \to Y$ of C weighted by W consists of a morphism $M \otimes X \to Y$ together with an element $w \in W(M)$, as depicted in figure 2(right).

Any bicategory can be turned into a 1-category by change of enrichment basis along the connected components functor $\pi_0 : \mathbf{Cat} \to \mathbf{Set}$. This operation quotients together 1-cells that are related by any 2-cell. (π_0 is right adjoint to the free functor $\mathbf{Set} \to \mathbf{Cat}$, and it is more common to change basis along the left adjoint, which instead quotients out only invertible 2-cells.) The 1-category $\pi_0^*(\mathbf{Para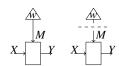}^W(C))$ has morphisms that are equivalence classes identifying all ways of making the cut in figure 2(right). This satisfies an important universal property: it is the symmetric monoidal category that results from freely extending C with a state $w : I \to X$ for each

Figure 2: Morphism in $\mathbf{Para}^W(C)$ (right) and its equivalence class in $\pi_0^*(\mathbf{Para}^W(C))$ (left).

element $w \in F(X)$, for all objects X [HT12][2].

Let C be a symmetric monoidal category. We define a symmetric lax monoidal functor $\mathbb{I} : \mathbf{Optic}(C) \to$ **Set**, called the **iteration functor** [Hed24]. On objects, we set $\mathbb{I}\binom{X}{X'} = \int^{M:C} C(I, M \otimes X) \times C(M \otimes X', M \otimes X)$. Given an element $(M, x_0, i) \in \mathbb{I}\binom{X}{X'}$ we call M the **state space**, $x_0 : I \to M \otimes X$ the **initial state** and $i : M \otimes X' \to M \otimes X$ the **iterator**.

Given an optic $f = (N, f, f') : \binom{X}{X'} \to \binom{Y}{Y'}$ in **Optic**(C), we get a function $\mathbb{I}(f) : \mathbb{I}\binom{X}{X'} \to \mathbb{I}\binom{Y}{Y'}$ given by taking (M, x_0, i) to the state space $M \otimes N$, the initial state $I \xrightarrow{x_0} M \otimes X \xrightarrow{M \otimes f} M \otimes N \otimes Y$, and the iterator

$$M \otimes N \otimes Y' \xrightarrow{M \otimes f'} M \otimes X' \xrightarrow{i} M \otimes X \xrightarrow{M \otimes f} M \otimes N \otimes Y$$

This can be easily checked to be functorial (see proposition A.1 in the appendix).

When $C = $ **Set** and similar cases, given an element $i = (M, (m_0, x_0), i) \in \mathbb{I}\binom{X}{X'}$ and a function $k : X \to X'$, we can define an infinite sequence $\langle k | i \rangle : X^\omega$ by the corecursive formula

$$\langle k | M, (m_0, x_0), i \rangle = x_0 : \langle k | M, i(m_0, k(x_0)), i \rangle$$

This defines a dinatural transformation $\langle - | - \rangle : \mathbb{K}\binom{X}{X'} \times \mathbb{I}\binom{X}{X'} \to X^\omega$. In the general case, this can be accomplished using the machinery of monoidal streams [LFR22].

We also have an evident laxator $\nabla : \mathbb{I}\binom{X}{X'} \times \mathbb{I}\binom{Y}{Y'} \to \mathbb{I}\binom{X \otimes Y}{X' \otimes Y'}$ defined up to symmetries by

$$(M, x_0, i) \nabla (M', x_0', i') = (M \otimes M', x_0 \otimes x_0', i_0 \otimes i_0')$$

The resulting symmetric monoidal category $\mathbf{Optic}^{\mathbb{I}}(C) = \pi_0^* \left(\mathbf{Para}^{\mathbb{I}}(\mathbf{Optic}(C)) \right)$ extends **Optic**(C) with states $I \to \binom{X}{X'}$ defined by elements of $\mathbb{I}\binom{X}{X'}$. A typical morphism is depicted at the end of [Hed24].

Given a symmetric monoidal category C, a **comorphism** $X \to Y$, which could also be called a **context** for morphisms $X \to Y$, is a state $I \to \binom{X}{X'}$ of **Optic**(C). When C is itself a category of optics, this is known as **double optics** and is a central idea of Bayesian open games [BHZ23]. These can be depicted as combs with 1 hole and bidirectional wires, or combs with 2 holes and only forwards wires.

Given a symmetric monoidal category C, a functor $\mathbf{Optic}(C) \to$ **Set** can be equivalently defined as a **Tambara comodule**: a profunctor $W : C \times C^{\mathrm{op}} \to$ **Set** equipped with a natural family of functions $W(M \otimes X, M \otimes Y) \to W(X, Y)$. This is a dualisation of the fundamental theorem of Tambara theory [PS08, CEG+24]. Given this data, a generalised comorphism $X \to Y$ can be defined as an element of $W\binom{X}{X'}$, that is, a state of $\mathbf{Optic}^W(C) = \pi_0^* \left(\mathbf{Para}^W(\mathbf{Optic}(C)) \right)$. This construction appears in [Hed23].[3]

Putting this together, we can define an **iteration context** for optics $\binom{X}{X'} \to \binom{Y}{Y'}$ as a (representable) state of $\mathbf{Optic}(\mathbf{Optic}^{\mathbb{I}}(C))$. This defines a functor $\mathbb{I}_{\mathrm{env}} : \mathbf{Optic}^2(C) \to$ **Set** depicted in figure 3(above), and by pulling the state variable of i through k we can define it equivalently by a coend over C rather than over $\mathbf{Optic}^{\mathbb{I}}(C)$:

$$\mathbb{I}_{\mathrm{env}} \left(\binom{X}{X'}, \binom{Y}{Y'} \right) \cong \int^{M, M':C} C(I, M \otimes X) \times C(M \otimes Y, M' \otimes Y') \times C(M' \otimes X', M \otimes X)$$

This can be equivalently depicted as a 3-hole comb in figure 3(below), and we can unroll this n steps to produce a $2n$-hole comb, including an ω-comb [LFR22] for the limiting case.

---

[2]Thanks to Nathan Corbyn for bringing this reference to our attention.

[3]The first author has been working on the theory of generalised contexts for several years, but has yet to find a compelling application outside of categorical cybernetics.

5 Bellman operators

A *Bellman operator* is a self-mapping of a function space of either state-value functions or state-action-value functions, which iteratively improves the estimation of values. For dynamic programming, the most basic Bellman operator $\mathfrak{B}_\pi = \mathfrak{B}_{\mathrm{val}}(-,\pi) : \mathbb{R}^S \to \mathbb{R}^S$ is defined by

$$\mathfrak{B}_\pi(V)(s) = \mathbb{E}_{(r,s')\sim t(s,\pi(s))}\left[r + \gamma V(s')\right]$$

The functional equation $V = \mathfrak{B}_\pi(V)$ is called a *Bellman equation*, and its solution V characterises the long-run values of the policy π. Provided S is finite and $0 < \gamma < 1$, $\mathfrak{B}_\pi$ is a contraction mapping on the supremum metric of $\mathbb{R}^S$, and therefore iterating $\mathfrak{B}_\pi$ from any initial estimate of V will converge to the unique solution of the Bellman equation.

Figure 3: Iteration contexts as states of **Optic**$^2(C)$ (above) and 3-hole combs (below).

In [HS23] we showed that $\mathfrak{B}_\pi$ has the form $\mathfrak{B}_\pi = \mathbb{K}(\ell_\pi)$, where $\ell_\pi = \binom{\ell_f}{\ell_b} : \binom{S}{\mathbb{R}} \to \binom{S}{\mathbb{R}}$ is the mixed optic in the category **Optic**$_{\mathrm{Kl}(D)}(\mathrm{Kl}(D), \mathrm{EM}(D))$ defined by $\ell_f : S \to D(S \times \mathbb{R})$ which maps $s \mapsto t(s, \pi(s))$ and $\ell_b : D(\mathbb{R}) \times \mathbb{R} \to \mathbb{R}$ which maps $r, v \mapsto \mathbb{E}[r] + \gamma v$. Here D is the finite support probability monad on **Set**, whose Eilenberg-Moore category is convex sets [Fri09], with $\mathrm{Kl}(D)$ acting on $\mathrm{EM}(D)$ via the embedding of free algebras as algebras $D : \mathrm{Kl}(D) \hookrightarrow \mathrm{EM}(D)$, which is a strong monoidal functor.

That is to say, if we represent a value function $V : S \to \mathbb{R}$ by a costate of optics $V : \binom{S}{\mathbb{R}} \to \binom{1}{1}$, then the costate $V \circ \mathcal{B}_\pi : \binom{S}{\mathbb{R}} \to \binom{1}{1}$ similarly represents $\mathfrak{B}(V) : S \to \mathbb{R}$. This is a refinement of the usual view of Bellman operators as endomorphisms of function spaces.

Strictly speaking it would be preferable to use a category of optics whose forward objects are finite sets and backward objects are complete metric spaces, using a suitable probability monad such as Kantorovich [FP19], in order to guarantee convergence in all cases. However we leave the details of this for future work. For the general theory of RL one can work with Markov categories [Fri20], and especially representable Markov categories [FGPR23] to handle the interplay between *distributions* and *samples*.

This type of Bellman operator updates an entire value function at once, as is most common in basic dynamic programming. However in reinforcement learning it is far more common to update a Q-matrix one entry at a time, with a sample determining which state-action pair is to have its value updated. Bellman operators of that form do not directly factor through $\mathbb{K}$. However we can fix this with a small change in perspective: we consider Bellman operators that return a *delta* or *change* to a Q-matrix, apply $\mathbb{K}$ to that, and then apply the change to obtain a new Q-matrix in the world of functions rather than optics. This same change of perspective is required anyway to describe deep RL where the delta is replaced with a cotangent vector, so it is an interesting observation that the category theory suggests the same for the more discrete setting of tabular RL.

One may ask whether $\mathfrak{B}_{\mathrm{pol}}$ arises also as the image under $\mathbb{K}$ of a certain optic, and the answer is negative. This was an ambiguous point in [HS23] where the policy improvement step could not be stated as an optic, even though both $\mathfrak{B}_{\mathrm{val}}$ and $\mathfrak{B}_{\mathrm{pol}}$ are treated as similar contraction operators in the dynamic programming literature. We can now give a more explicit answer; $\mathfrak{B}_{\mathrm{pol}}$ involves two coevaluation maps (λ), so it is not in the image of an optic under the continuation functor (see figure 4).
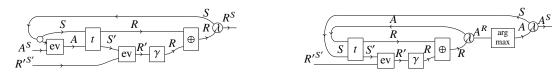
Figure 4: $\mathfrak{B}_{\text{val}}$ (left) and $\mathfrak{B}_{\text{pol}}$ (right) in the free autonomisation of **Set** [Del20] (see also [BS10]).

5.1 Parametric Bellman operators

Since the category **Lens** = **Optic**(**Set**) is enriched in **Set**, we can form its category of externally parametrised morphisms, **Para$_{\text{Set}}$(Lens)**. A morphism $\binom{X}{X'} \to \binom{Y}{Y'}$ of this category consists of a parameter set P, a forwards pass function $P \times X \to Y$ and a backwards pass function $P \times X \times Y' \to X'$.

The simplest Bellman operator for RL, the one for SARSA, is a morphism in this category of type $\mathfrak{B} : \binom{1}{S \times A \times \mathbb{R}} \to \binom{S \times A}{\mathbb{R}}$, with parameter set $\Upsilon = S \times A \times \mathbb{R} \times S \times A$, where the backward pass function is $r, v \mapsto r + \gamma v$ (figure 5).

We lift the functor $\mathbb{K} : \textbf{Lens}^{\text{op}} \to \textbf{Set}$ to the functor **Para$_{\text{Set}}$($\mathbb{K}$)** : **Para$_{\text{Set}}$(Lens$^{\text{op}}$)** $\to$ **Para$_{\text{Set}}$(Set)**. Applying this functor to $\mathfrak{B}$ results in a function

$$\textbf{Para}_{\textbf{Set}}(\mathbb{K})(\mathfrak{B}) : \Upsilon \times \mathbb{R}^{S \times A} \to S \times A \times \mathbb{R}$$
$$((s, a, r, s', a'), Q) \mapsto (s, a, r + \gamma Q(s', a'))$$

Figure 5: Target computation in SARSA as a parametrised lens, and its $\mathbb{K}$-image in **Set**.

Here we very informally think of $S \times A \times \mathbb{R}$ as a 'discrete cotangent vector' at $Q \in \mathbb{R}^{S \times A}$. Slightly more precisely, the actual cotangent space is $\mathbb{R}^{S \times A}$, with $S \times A \times \mathbb{R}$ representing a scaled basis vector via the embedding $S \times A \times \mathbb{R} \hookrightarrow \mathbb{R}^{S \times A}$, $(s, a, r) \mapsto$ $(s', a') \mapsto \begin{cases} r & \text{if } s = s', a = a' \\ 0 & \text{otherwise} \end{cases}$. This differential geometry perspective is heavily inspired by Myers' categorical systems theory [Mye23] and related unpublished work in progress of Capucci, and we leave it for future work to make its application to reinforcement learning precise.

We do not provide a general definition of Bellman operators, and consider this a representative 'definition by example'. In general, Bellman operators will be optics from value functions to deltas of value functions, parametrised by a *sample* which is data received from the environment.

In methods where the sample requires the operator to make use of the continuation only once, this parametric operator can be represented as a lens. SARSA is an example of this, where the usage of Q by the target is *linear* in the sense of linear type theory. Setting aside the convergence properties of the Bellman operator, we treat it from this point on as a morphism $G : \Upsilon \times \mathbb{R}^S \to \mathbb{R}^S$ in **Set**. This morphism becomes a central part of our formalization of an RL model, explained next.

6 Models, agents and environments

A model for an RL method contains the data to generate the policy from certain inner parameters and the data to update those parameters based on bootstrapping and/or samples. It matches the structure of a lens from model parameters to agent interface, which we annotate in figure 6(a).

The forward map uses parameters of the method to generate a policy for the agent's interaction with the environment. In Q-learning for example, which is a TD method (figure 6(b)), $P : \mathbb{R}^{S \times A} \to$

$(\mathcal{P}A)^S$ takes the current Q-table $Q : \mathbb{R}^{S \times A}$ and returns the greedy policy $\pi : (\mathcal{P}A)^S$ defined by $\pi(s) = \mathrm{argmax}_a\, Q(s,a)$.

The backward map takes the return from the agent and the current parameters to generate an update target as a (often discrete) cotangent vector for the parameters. The return usually takes the form of some product of types S (states), A (actions) and $\mathbb{R}$ (rewards). In SARSA (figure 6(b)), G takes a sample (s,a,r,s',a') (right input) and the bootstrapped Q-table (upper input) to calculate the target $r + \gamma Q(s',a')$, which is the direction of the cotangent vector at (s,a). In DP (figure 6(d)), there is no return from the agent, and the update target is the output of the Bellman operator $\mathfrak{B}_{\mathrm{val}}$ as a section of the cotangent bundle: for every state $s \in S$, $\mathfrak{B}_{\mathrm{val}}(V)(s)$ defines the direction that $V(s)$ must change to.

Certain DP methods like GPI or asynchronous DP [Ber82] benefit from treating the two Bellman operators as separate backward morphisms (figure 6(e)). This also illustrates **actor-critic** (AC) methods [KT01] (figure 6(f)), which keep two separate functions, the *actor* $P : \Theta \times S \to DA$, similar to a policy gradient network, and the *critic* $V : \Omega \to S \to \mathbb{R}$, which learns a baseline value function. This learned baseline serves to reduce the high variance of data samples seen in PG methods. Being a deep RL method, the actor map $P : \theta \mapsto \pi_\theta$ is a neural network, and the associated backward map $\mathcal{L}_{\mathrm{ac}} : \Theta \times \Omega \times SA\mathbb{R} \to \Delta\Theta$ is defined by the improvement of expected return $(\theta, \omega, s, a, r) \mapsto (r - V_\omega(s))\nabla_\theta \log \pi_\theta(s,a)$ with a baseline given by V_ω. The critic map $V : \omega \mapsto V_\omega$ has as the backward map $\mathcal{L}_{\mathrm{cr}} : \Omega \times SA\mathbb{R}S \to \Delta\Omega$ the reduction of policy update variance $(\omega, s, a, r, s') \mapsto r + \gamma V_\omega(s') - V_\omega(s)$.
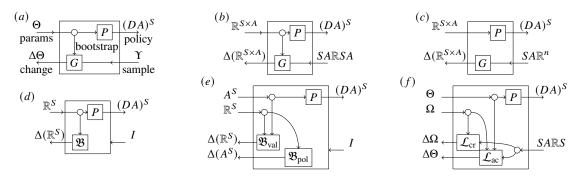

Figure 6: Lenses for (*a*) a generic RL model, (*b*) TD, (*c*) MC, (*d*) DP, (*e*) GPI and (*f*) Actor-Critic methods. The drawing of the backward map as a morphism with an input from the top is merely a stylistic choice, where it should be understood as a morphism with two inputs, the bootstrap and the sample.

This model lens embeds into **Optic**(C) which is extended to **Optic**$^{\mathbb{I}}(C)$ by an iteration functor $\mathbb{I}$ which we define next. The left interface to the model optic is closed by two pieces of data: An initial state $q_0 : I \to M \otimes \Theta$ and an update rule $i : M \otimes \Delta\Theta \to M \otimes \Theta$ that acts as the iterator. In gradient-free methods, the update is generally pointwise as in (2). Conversely, gradient-based methods use neural network optimizers like stochastic gradient descent, Adam and other variations as the update rule [CGG⁺22].

The right interface of a model parametrises an agent, which is a morphism $\binom{S}{I} \to \binom{A}{F}$ in **Para**(**Optic**(C)). This parametrised morphism will itself interact with an environment that is an iteration context. The coend in the environment is taken over states of the Markov chain.

Offline methods, unlike online methods, interact with the agent only in a trivial way by showing it experiential samples. This is shown by the types $M = S \times A \times F$ and $M' = I$, by which the continuation ignores the agent's action and just projects the action and feedback as a response to the agent. Moreover, the iterator type $C(M' \otimes I, M \otimes S) \cong C(I, M \otimes S)$ coincides with the initial state, which reflects the fact that the environment samples experiences (s,a,f) from a distribution defined by a dataset (figure 8(a,b)).

To clarify the interplay between these the three structures described in this section, we look at the
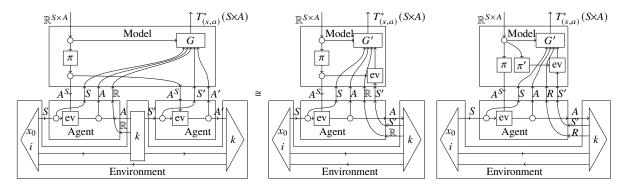
Figure 7: SARSA is on-policy (left two). Q-learning is off-policy (right).

role played by internal and external policies in on- and off-policy methods. First, figure 7(left) shows the full representation of SARSA. It consists of a model optic parametrising two copies of an agent that are composed with a 2-hole environment. The policy evaluated by both instances is the same, and the return to the model consists of (s, a, r) from the first agent optic and (s', a') from the second. SARSA is an on-policy method, as the policy deployed to obtain $a' \sim \pi(s')$ is the same as the one used to compute the first action $a \sim \pi(s)$. Calculating the target G from the sample (s, a, r, s', a') is equivalent to calculating G from (s, a, r, s') and its internal policy π, even though the model does not know the environment's dynamics k. This is why the same method can be equivalently specified by the middle diagram.

On the other hand, Q-learning (figure 7(right)) is an off-policy method, because the last action is computed by an internal policy $\pi' = \text{argmax}$ different from the one being deployed.

Figure 8: Online (a) and offline (b) RL environments. Contextual (c) and multi-armed (d) bandit environments. Omitted arrows are the unit. The offline continuation is a projection p of $A \times F$.

6.1   Prediction and bandit problems

The presented framework handles RL **prediction** problems for free in all the previous methods by trivialising the set $A = 1$, which pinpoints the idea that *a prediction algorithm is a control algorithm where there's no choice of actions*. For example, MC prediction of the long-term value of states from n-long episodes becomes an optic $\binom{\mathbb{R}^S}{T_s^* S} \to \binom{I}{\mathbb{SR}^n}$, and 1-TD prediction becomes $\binom{\mathbb{R}^S}{T_s^* S} \to \binom{I}{\mathbb{SRS}}$. The forward maps for both are trivial since the agent has no policy to execute, perhaps better called **observer** rather than agent here. The corresponding environments have the type of a MRP.

Moreover, **bandit problems** emerge by trivialising $M' = I$ (figure 8(c,d)). In particular, contextual bandits involve finding the best action in A associated to a particular state in M for which only partial information of type S is given, yielding feedback in F. This action does not affect further distributions of states, so the object between the continuation and the update rule is trivial. Multi-armed bandit problems are a further special case, characterized by environments whose only non-trivial morphism is the continuation $k : A \to F$.

References

[AAG03]   Michael Abbott, Thorsten Altenkirch & Neil Ghani (2003): *Categories of containers*. In: *Proceedings of FoSSACS 2003*, Lecture Notes in Computer Science, Springer.

[Bar93]   E. Barnard (1993): *Temporal-difference methods and Markov models*. IEEE Transactions on Systems, Man, and Cybernetics 23(2), pp. 357–365, doi:10.1109/21.229449.

[Ber82]   D. Bertsekas (1982): *Distributed dynamic programming*. IEEE Transactions on Automatic Control 27(3), pp. 610–616, doi:10.1109/tac.1982.1102980.

[Ber19]   Dimitri P. Bertsekas (2019): *Reinforcement Learning and Optimal Control*. Athena Scientific optimization and computation series, Athena Scientific.

[BHS23]   Dylan Braithwaite, Jules Hedges & Toby St Clere Smithe (2023): *The compositional structure of Bayesian inference*. In: *Proceedings of Mathematical Foundations of Computer Science 2023*, Leibniz Proceedings in Informatics 272.

[BHZ23]   Joe Bolt, Jules Hedges & Philipp Zahn (2023): *Bayesian open games*. Compositionality 5(9).

[BJI⁺17]   Nicola Botta, Patrik Jansson, Cezar Ionescu, David R. Christiansen & Edwin Brady (2017): *Sequential decision problems, dependent types and generic solutions*. Logical Methods in Computer Science 13(1).

[BS10]   John Baez & Mike Stay (2010): *Physics, topology, logic and computation: A Rosetta Stone*. In: *New structures for physics*, Springer.

[CEG⁺24]   Bryce Clarke, Derek Elkins, Jeremy Gibbons, Fosco Loregian, Bartosz Milewski, Emily Pillmore & Mario Román (2024): *Profunctor optics, a categorical update*. Compositionality 6(1).

[CG23]   Matteo Capucci & Bruno Gavranović (2023): *Actegories for the working amthematician*. ArXiv:2203.16351.

[CGG⁺22]   Geoffrey Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson & Fabio Zanasi (2022): *Categorical foundations of gradient-based learning*. In: *Proceedings of ESOP 2022*, Lecture Notes in Computer Science 13240.

[CGHR22]   Matteo Capucci, Bruno Gavranović, Jules Hedges & Eigil Rischel (2022): *Towards foundations of categorical cybernetics*. In: *Proceedings of Applied Category Theory 2021*, Electronic Proceedings in Theoretical Computer Science 372.

[Dal19]   Davidad Dalrymple (2019): *Dioptics: a common generalization of open games and gradient-based learners*. Unpublished paper available at https://research.protocol.ai/publications/dioptics-a-common-generalization-of-open-games-an

[Del20]   Antonin Delpeuch (2020): *Autonomization of Monoidal Categories*. In: *Proceedings of Applied Category Theory 2019*, Electronic Proceedings in Theoretical Computer Science.

[FGPR23]   Tobias Fritz, Tomáš Gonda, Paolo Perrone & Eigil Fjeldgren Rischel (2023): *Representable Markov categories and comparison of statistical experiments in categorical probability*. Theoretical computer science 961.

[FP19]   Tobias Fritz & Paolo Perrone (2019): *A probability monad as the colimit of spaces of finite samples*. Theory and applications of categories 34(7), pp. 170–220. arXiv:1712.05363.

[Fri09]   Tobias Fritz (2009): *Convex spaces I: Definitions and examples*. ArXiv:0903.5522.

[Fri20]   Tobias Fritz (2020): *A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics*. Advances in Mathematics 370, p. 107239, doi:10.1016/j.aim.2020.107239. Available at http://arxiv.org/abs/1908.07021. ArXiv: 1908.07021.

[Gav22]   Bruno Gavranović (2022): *Space-time tradeoffs of lenses and optics via higher category theory*. ArXiv: 2209.09351.

[Gav24]   Bruno Gavranović (2024): *Fundamental components of deep learning: A category-theoretic approach*. Ph.D. thesis, University of Strathclyde.

[Ghr08]   Robert Ghrist (2008): *The persistent topology of data*. Bulletin of the American Mathematical Society 45.

[GHWZ18]   Neil Ghani, Jules Hedges, Viktor Winschel & Philipp Zahn (2018): *Compositional game theory*. In: *Proceedings of Logic in Computer Science 2018*, ACM.

[Hed23]   Jules Hedges (2023): *The game semantics of game theory*. In: *Samson Abramsky on Logic and Structure in Computer Science and Beyond*, Outstanding contributions to logic 25, Springer.

[Hed24]   Jules   Hedges   (2024):   *Iteration   with   optics*.   Blog   post   available   at   <https://cybercat.institute/2024/02/22/iteration-optics/>.

[HS19]   Linde Hansen & Mike Shulman (2019): *Constructing symmetric monoidal bicategories functorially*. ArXiv:1910.09240.

[HS23]   Jules Hedges & Riu Rodríguez Sakamoto (2023): *Value iteration is optic composition*. In: *Proceedings of Applied Category Theory 2022*, Electronic Proceedings in Theoretical Computer Science 380.

[HT12]   Claudio Hermida & Robert D Tennent (2012): *Monoidal indeterminates and categories of possible worlds*. Theoretical computer science 430.

[Kel82]   G.M. Kelly (1982): *Basic concepts of enriched category theory*. Lecture Notes in Mathematics 64, Cambridge University Press.

[KLM96]   L. P. Kaelbling, M. L. Littman & A. W. Moore (1996): *Reinforcement Learning: A Survey*. Journal of Artificial Intelligence Research 4, pp. 237–285, doi:10.1613/jair.301.

[KT01]   Vijay Konda & John Tsitsiklis (2001): *Actor-Critic Algorithms*. Society for Industrial and Applied Mathematics 42.

[LFR22]   Elena di Lavore, Giovanni de Felice & Mario Román (2022): *Monoidal streams for dataflow programming*. In: *Proceedings of Logic in Computer Science 2022*, ACM.

[Lor21]   Fosco Loregian (2021): *(Co)end calculus*. Cambridge University Press.

[MKS+13]   Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra & Martin Riedmiller (2013): *Playing Atari with Deep Reinforcement Learning*, doi:10.48550/ARXIV.1312.5602.

[Mye23]   David Jaz Myers (2023): *Categorical systems theory*. Draft book.

[NS24]   Nelson Niu & David Spivak (2024): *Polynomial functors: A mathematical theory of interaction*. Draft book.

[PS08]   Craig Pastro & Ross Street (2008): *Doubles for monoidal categories*. Theory and applications of categories 21(4).

[Ril18]   Mitchell Riley (2018): *Categories of optics*. ArXiv:1809.00738.

[SB20]   Richard S. Sutton & Andrew G. Barto (2020): *Reinforcement Learning: An Introduction*. MIT Press.

[Smi23]   Toby St Clere Smithe (2023): *Mathematical foundations for a compositional account of the Bayesian brain*. Ph.D. thesis, University of Oxford.

[SMSM99]   Richard S. Sutton, David McAllester, Satinder Singh & Yishay Mansour (1999): *Policy gradient methods for reinforcement learning with function approximation*. In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, MIT Press, Cambridge, MA, USA, pp. 1057–1063.

[SSPS21]   David Silver, Satinder Singh, Doina Precup & Richard S. Sutton (2021): *Reward is enough*. Artificial Intelligence 299, p. 103535, doi:10.1016/j.artint.2021.103535.

[SSS+17]   David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel & Demis Hassabis (2017): *Mastering the game of Go without human knowledge*. Nature 550.

[Sut95]   Richard S. Sutton (1995): *Generalization in reinforcement learning: successful examples using sparse coarse coding*. In: *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, MIT Press, pp. 1038–1044.

[VBC+19]   Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Jun-young Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps & David Silver (2019): *Grandmaster level in StarCraft II using multi-agent reinforcement learning*. Nature 575.

[VC22]   André Videla & Matteo Capucci (2022): *Lenses for composable servers*. ArXiv:2203.15633.

[Ver23]   Pietro Vertechi (2023): *Dependent optics*. In: *Proceedings of Applied Category Theory 2022*, EPTCS.

[WD92]   Christopher J. C. H. Watkins & Peter Dayan (1992): *Q-learning*. Machine Learning 8(3-4), pp. 279–292, doi:10.1007/bf00992698.

A   Appendix

Proposition A.1. *The iterator* $\mathbb{I} : \mathbf{Optic}(C) \to \mathbf{Set}$ *is functorial.*

Proof. Let $f = (N, f, f') : \binom{X}{X'} \to \binom{Y}{Y'}$ and $g = (P, g, g') : \binom{Y}{Y'} \to \binom{Z}{Z'}$ be two morphisms in $\mathbf{Optic}(C)$. Preservation of identity is shown by:

$$\mathbb{I}(I, 1_X, 1_{X'}) : (M, x_0, i) \mapsto (M \otimes I, x_0; (I \otimes 1_X), (M \otimes 1_{X'}); i; (M \otimes 1_X))$$

Preservation of composition is shown by the isomorphic images of $\mathbb{I}(N, f, f'); \mathbb{I}(P, g, g')$, which maps $(M, x_0, i) : \mathbb{I}\binom{X}{X'}$ to the state space $M \otimes N \otimes P$, the initial state $I \xrightarrow{x_0} M \otimes X \xrightarrow{M \otimes f} M \otimes N \otimes Y \xrightarrow{M \otimes N \otimes g} M \otimes N \otimes P \otimes Z$ and the iterator

$$M \otimes N \otimes P \otimes Z' \xrightarrow{M \otimes N \otimes g'} M \otimes N \otimes Y' \xrightarrow{M \otimes f'} M \otimes X' \xrightarrow{i} M \otimes X \xrightarrow{M \otimes f} M \otimes N \otimes Y \xrightarrow{M \otimes N \otimes g} M \otimes N \otimes P \otimes Z$$

which defines the element in $\mathbb{I}\binom{Z}{Z'}$, and $\mathbb{I}(N \otimes P, (f; N \otimes g), (N \otimes g'; f'))$, which maps (M, x_0, i) to the same state space, the initial state $x_0; (M \otimes (f; N \otimes g))$, and the iterator $(M \otimes (N \otimes g'); f'); i; (M \otimes (f; N \otimes g))$. $\square$