Value iteration is optic composition

Jules Hedges Riu Rodríguez Sakamoto

Dynamic programming is a class of algorithms used to compute optimal control policies for Markov decision processes. Dynamic programming is ubiquitous in control theory, and is also the foundation of reinforcement learning. In this paper, we show that value improvement, one of the main steps of dynamic programming, can be naturally seen as composition in a category of optics, and intuitively, the optimal value function is the limit of a chain of optic compositions. We illustrate this with three classic examples: the gridworld, the inverted pendulum and the savings problem. This is a first step towards a complete account of reinforcement learning in terms of parametrised optics.

1 Introduction

In this paper we describe basic concepts of dynamic programming in terms of categories of optics. The class of models we consider are discrete-time Markov decision processes, aka. discrete-time controlled Markov chains. There are classical methods of computing optimal control policies, underlying much of both classical control theory and modern reinforcement learning, known collectively as *dynamic programming*. These are based on two operations that can be interleaved in many different ways: *value improvement* and *policy improvement*. The central idea of this paper is the slogan *value improvement is optic precomposition*, or said differently, *value improvement is a representable functor on optics*.

Given a control problem with state space X, a *value function* $V : X \to \mathbb{R}$ represents an estimate of the long-run payoff of following a policy starting from any state, and can be equivalently represented as a costate $V : \binom{X}{\mathbb{R}} \to I$ in a category of optics. Every control policy π also induces an optic $\lambda(\pi) : \binom{X}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$. The general idea is that the forwards pass of the optic is a morphism $X \to X$ describing the dynamics of the Markov chain given the policy, and the backwards pass is a morphism $X \otimes \mathbb{R} \to \mathbb{R}$ which given the current state and the *continuation payoff*, describing the total payoff from all future stages, returns the total payoff for the current stage given the policy, plus all future stages.

Given a policy π and a value function $V : \binom{X}{\mathbb{R}} \to I$, the costate $\binom{X}{\mathbb{R}} \xrightarrow{\lambda(\pi)} \binom{X}{\mathbb{R}} \xrightarrow{V} I$ is a closer approximation of the value of π. This is called *value improvement*. Iterating this operation

$$\dots \binom{X}{\mathbb{R}} \xrightarrow{\lambda(\pi)} \binom{X}{\mathbb{R}} \xrightarrow{\lambda(\pi)} \binom{X}{\mathbb{R}} \xrightarrow{V} I$$

converges efficiently to the true value function of the policy π.

Replacing π with a new policy that is optimal for its value function is called *policy improvement*. Repeating these steps is known as *policy iteration*, and converges to the optimal policy and value function.

Alternatively, instead of repeating value improvement until convergence before each step of policy improvement, we can also alternate them, giving the composition of optics

$$\dots \binom{X}{\mathbb{R}} \xrightarrow{\lambda(\pi_2)} \binom{X}{\mathbb{R}} \xrightarrow{\lambda(\pi_1)} \binom{X}{\mathbb{R}} \xrightarrow{V} I$$

where each policy π_i is optimal for the value function to the right of it. This is known as *value iteration*, and also converges to the optimal policy and value function. For an account of convergence properties of these algorithms, classic textbooks are [32, Sec.6], [5, Ch.1].

Submitted to:

In this paper we illustrate this idea, using mixed optics to account for the categorical structure of transitions in a Markov chain and the convex structure of expected payoffs, which typically form the kleisli and Eilenberg-Moore categories of a probability monad. This paper is partially intended as an introduction to dynamic programming for category theorists, focussing on illustrative examples rather than on heavy theory.

1.1 Related work

The precursor of this paper was early work on value iteration using open games [21]. The idea originally arose around 2016 during discussions of the first author with Viktor Winschel and Philipp Zahn. An early version was planned as a section of [25] but cut partly for page limit reasons, and partly because the idea was quite uninteresting until it was understood how to model stochastic transitions in open games [6] via optics [33]. In this paper we have chosen to present the idea without any explicit use of open games, both in order to clarify the essential idea and also to bring it closer to the more recent framework of categorical cybernetics [14], which largely subsumes open games [10]. (Although, actually using this framework properly is left for future work.)

A proof-of-concept implementation of value iteration with open games was done in 2019 by the first author and Wolfram Barfuss[1], implementing a model from [3] - a model of the social dilemma of emissions cuts and climate collapse as a stochastic game, or jointly controlled MDP - and verifying it against Barfuss' Matlab implementation. A far more advanced implementation of reinforcement learning using open games was developed recently by Philipp Zahn, currently closed-source, and was used for the paper [15].

The most closely related work to ours is [26], which formulates MDPs in terms of F-lenses [35] of the functor $\mathrm{BiKl}(C \times -, \Delta(\mathbb{R} \times -))^{\mathrm{op}}$, where $C \times -$ is the reader comonad and $\Delta(\mathbb{R} \times -)$ is a probability monad over actions with their expected value. A MDP there is a lens from states and potential state changes and rewards to the agents observation and input $\binom{X}{\Delta(X \times \mathbb{R})} \to \binom{O}{I}$. Our approach differs in two ways. We firstly assume that the readout function is the identity, as we are not dealing with partial observability [1]. Secondly, we specify a concrete structure of the backwards update map $f^* : X \times I \to \Delta(X \times R)$, which allows us to rearrange the interface of this lens from policies to value functions. Doing so opens up the possibility of composing these lenses sequentially, which is the heart of the dynamic programming approach explored in this paper.

Another approach is to model MDPs as coalgebras from states to rewards and potential transitions, as done by Feys et al. [16]. They observe that the Bellman optimality condition for value iteration is a certain coalgebra-to-algebra morphism. We believe that this approach is orthogonal to ours, and both could potentially be done simultaneously. We discuss this in the further work section.

A series of papers by Botta et al (for example [7]) formulates dynamic programming in dependent type theory, accounting in a serious way for how different actions can be available in different states, a complication that we ignore in this paper. It may be possible to unify these approaches using dependent optics [8, 40].

Finally, [2] builds a category of signal flow diagrams, a widely used tool in control theory. Besides the common application to control theory there is little connection to this paper. In particular, time is implicit in their string diagrams, meaning their models have continuous time, whereas our approach is inherently discrete time. Said another way, composition in their category is 'space-like' whereas ours is 'time-like' - their morphisms are (open) systems whereas ours are processes.

[1] Source currently available at `https://github.com/jules-hedges/open-games-hs/blob/og-v0.1/src/OpenGames/Examples/EcologicalPublicGood/EcologicalPublicGood.hs`

2 Dynamic programming

2.1 Markov Decision Processes

A *Markov decision process* (MDP) consists of a state space X, an action space A, a state transition function $f : X \times A \to X$, and a utility or reward function $U : X \times A \to \mathbb{R}$. The state transition function is often taken to be stochastic, that is, to be given by probabilities $f(x' \mid x, a)$. In the stochastic case the utility function can be taken without loss of generality to be an expected utility function. We imagine actions to be chosen by an agent, who is trying to *control* the Markov chain with the objective of optimising the long-run reward.

A *policy* for an MDP is a function $\pi : X \to A$, which can also be taken to be either deterministic or stochastic. The type of policies encodes the Markov property: the choice of action depends only on the current state, and may not depend on any memory of past states.

Given an initial state $x_0 \in X$, a policy π determines (possibly stochastically) a sequence of states

$$x_0, \quad x_1 = f(x_0, \pi(x_0)), \quad x_2 = f(x_1, \pi(x_1)), \quad \dots$$

The total payoff is given by an infinite geometric sum of individual payoffs for each transition:

$$V_\pi(x_0) = \sum_{k=0}^{\infty} \beta^k U(x_k, \pi(x_k)) \tag{1}$$

where $0 < \beta < 1$ is a fixed *discount factor* which balances the relevance of present and future payoffs. (There are other methods of obtaining a single objective from an infinite sequence of transitions, such as averaging, but we focus on discounting in this paper.) A key idea behind dynamic programming is that this geometric sum can be equivalently written as a telescoping sum:

$$V_\pi(x_0) = U(x_0, \pi(x_0)) + \beta(U(x_1, \pi(x_1)) + \beta(U(x_2, \pi(x_2) + \cdots)))$$

The *control problem* is to choose a policy π in order to maximise (the expected value of) $V_\pi(x_0)$. In terms of decision theory, we assume that the agent choosing the policy operates under rational behaviour. Continuous and independent preferences of outcome implies by the von Neumann-Morgenstern expected utility theorem that the utility function has as codomain the reals.

2.2 Deterministic dynamic programming

In dynamic programming, the agent's objective of maximizing the overall utility can be divided into two orthogonal goals: to determine the value of a given policy π (which we call the *value improvement* step), and to determine the optimal policy π^* (the *policy improvement* step). Bellman's equation is used as an update rule for both:

$$\textbf{Value improvement:} \qquad V'(x) = U(x, \pi(x)) + \beta V(f(x, \pi(x))) \tag{2}$$

$$\textbf{Policy improvement:} \qquad \pi'(x) = \arg\max_{a \in A} U(x, a) + \beta V(f(x, a)) \tag{3}$$

A Bellman optimality condition on the other hand determines the fixpoint of this update rule, and is met when $V' = V$ and $\pi' = \pi$ respectively.

The update rule (2) is the discounted sum (1) where the stream of states is co-recursively fixed by the policy π and transition function f. The co-recursive structure refers to the calculation of the utility of a

state x, where one needs the utility of the *next* state, while in a recursive structure, x needs the *previous* state, starting from an initial state as a base case.

Two classical algorithms use these two steps differently: Policy iteration iterates value improvement until the current policy value is optimal before performing a policy improvement step, and value iteration interleaves both steps one after another.

In policy iteration, a initial value function is chosen (usually $V(x) = 0$), and a randomly chosen policy π is evaluated by (2) repeatedly until the value reaches a fixpoint, which is assured by the contraction mapping [13]. Once V reaches (or in practice gets close to) a fixpoint $V' = V$ or another convergence condition, the policy improvement step (3) chooses a greedy policy as an improvement to π.

A q-function or *state-action value function* $q_\pi : X \times A \to \mathbb{R}$ describes the value of being in state x and then taking action a, assuming that subsequent actions are taken by the policy π

$$q_\pi(x,a) = U(x,a) + \beta V(f(x,a)) \tag{4}$$

The *policy improvement theorem* [4] states that if a pair of deterministic policies $\pi, \pi' : X \to A$ satisfies for all $x \in X$

$$q_\pi(x, \pi'(x)) \geq V_\pi(x)$$

then $V_{\pi'}(x) \geq V_\pi(x)$ for all $x \in X$.

The optimal policy π^*, if it exists, is the policy which if followed from any state, generates the maximum value. This is a Bellman optimality condition which fuses the two steps (2), (3):

$$V_{\pi^*}(x) = \max_{a \in A} U(x,a) + \beta V_{\pi^*}(f(x,a)) \tag{5}$$

Value iteration is a special policy iteration algorithm insofar it stops the update rule for value improvement to one step, by truncating the sum (1) to the first summand. Moreover, it introduces the value improvement step implicitly in the policy improvement, which assigns a value to states

$$V'(x) = \max_{a \in A} U(x,a) + \beta V(f(x,a))$$

while the policy in each iteration is still recoverable as

$$\pi'(x) = \arg\max_{a \in A} U(x,a) + \beta V(f(x,a))$$

2.3 Stochastic dynamic programming

Stochasticity can be introduced in different places in a MDP:

1. in the policy $\pi : X \to \Delta A$, where the probability of the policy π taking action a in a state x is now notated $\pi(a \mid x)$.
2. in the transition function $f : X \times A \to \Delta X$ and potentially the reward function $U : X \times A \to \Delta\mathbb{R}$ independently.
3. usually the reward is included inside the transition function $f : X \times A \to \Delta(X \times \mathbb{R})$, allowing correlated next states and rewards. This is relevant when the reward is morally from the next state, rather than the current state and action. If the reward were truly from the current state and action, the transition function can be decomposed into a function $f : X \times A \to \Delta X \times \Delta\mathbb{R}$.

In this section we assume for simplicity that Δ is the finite support distribution monad, although the equations in the following can be formulated for arbitrary distributions by replacing the sum with an appropriate integral.

The policy value update rule (2) becomes stochastic, and adopts a slightly different form depending on which part of the MDP is stochastic. For the cases 1. and 2.:

$$V'(x) = \sum_a \pi(a \mid x)(U(x,a) + \beta V(f(x,a))) \tag{6}$$

$$V'(x) = \sum_r U(r \mid x,a)r + \sum_{x'} f(x' \mid x,a)\beta V(x') \tag{7}$$

In the most general case, that is 1. together with 3.:

$$V'(x) = \sum_{a \in A} \pi(a \mid x) \sum_{x',r} f(x',r \mid x,a)(r + \beta V(x')) \tag{8}$$

(Note that the sum over r is over the support of $f(- \mid x,a)$, which we assume here to be finite, although in general it can be replaced with an integral.)

The policy improvement theorem holds in the stochastic setting [38, Sec.4.2] by defining

$$q_\pi(s, \pi'(s)) = \sum_a \sigma(a \mid s) q_\pi(s,a)$$

2.4 Gridworld example

A classic example in reinforcement learning is the Gridworld environment, where an agent moves in the four cardinal directions in a rectangular grid. States of this finite MDP correspond to the positions that the agent can be in.

Assume that all transitions and policies are deterministic, and that the transition function prevents the agent from moving outside the boundary. Suppose that the environment rewards 0 value for all states except the top left corner, where the reward is 1 (see figure 1).

Starting with a policy which moves upwards in all states and a value function which rewards 1 only in the top left corner, a policy iteration algorithm would improve the value of the current policy until converging to the optimal values in the leftmost column, before updating the policy, while a value iteration algorithm would update the value function and also update the policy.

Take the finite set of positions as the state space X, and $A = \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$ as the action space.

This example can be made stochastic if we add stochastic policies like ε-greedy, where the action that the agent takes is the one with maximum value with probability $1 - \varepsilon$ and a random one with probability ε. Another way is for the transition function to be stochastic, for example with a wind current that shifts the next state to the right with some probability ε.

2.5 Inverted pendulum example

A task that illustrates a continuous state space MDP is the control of a pendulum balanced over a cart, which can be described in continuous-time exactly by two non-linear differential equations [17, Example 2E]:

$$\begin{aligned} (M+m)\ddot{y} + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2 \sin\theta &= a \\ mL\ddot{y}\cos\theta + mL^2\ddot{\theta} - mLg\sin\theta &= 0 \end{aligned}$$

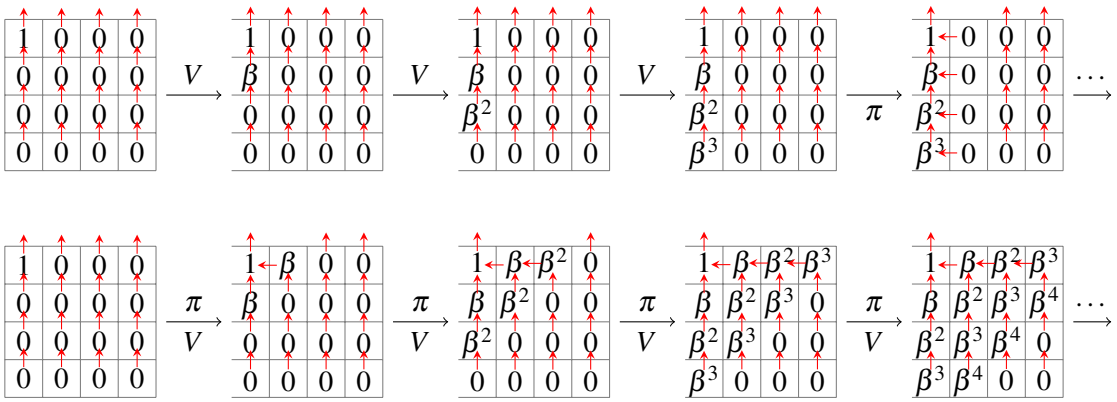


Figure 1: Difference between policy iteration (above) and value iteration (below). The numbers in the cells are state values and the red arrows are the directions dictated by the policy at each stage. The arrows between grids indicate what kind of update the algorithm does, either value improvement (V) or policy improvement (π). Notice how policy iteration performs value improvement three times before updating the policy, whereas value iteration improves the value and the policy at each stage.

where M is the mass of the cart, m the mass of the pendulum, L the length of the pendulum, θ the angle of the pendulum with respect to the upwards position, y the carts horizontal position, g the gravitational constant and a is our control function (usually denoted u). We rewrite the state variables as $x = [y, \dot{y}, \theta, \dot{\theta}]^\top$.

Sampling the trajectory of continuous-time dynamics $\frac{\mathrm{d}}{\mathrm{d}t}x(t) = f(x(t))$ by $x_k = x(k\Delta t)$, one can define the discrete-time propagator $F_{\Delta t}$ by

$$F_{\Delta t}(x(t)) = x(t) + \int_t^{t+\Delta t} f(x(\tau))\mathrm{d}\tau$$

which allows to model the system with $x_{k+1} = F_{\Delta t}(x_k)$.

A more common approach is to observe that the system of equations $\dot{x} = A(x) + B(x)a$ with A and B being non-linear functions of the state space, can be *linearized* near a (not necessarily stable) equilibrium state, like the pendulum being in the upwards position. There we can assume certain approximations like $\cos\theta \approx 1$ and $\sin\theta \approx \theta$, as well as small velocities leading to negligible quadratic terms $\dot{\theta}^2 \approx 0$ and $\dot{y}^2 \approx 0$. This linearization around a fixpoint allows for the expression $\dot{x} = Ax(t) + Ba(t)$, where the matrix A and vector B are constants given by

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{(M+m)g}{ML} & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 \\ \frac{1}{M} \\ 0 \\ -\frac{1}{ML} \end{pmatrix}$$

If we assume that the observation of the pendulum angle and cart position is discretized in time, an a priori time-discretization of this model using Euler approximation follows $x_{k+1} = x_k + \Delta t(Ax_k + Ba_k)$, with the same constants, where k indexes time steps. Therefore we can say that the time-discretized, linearized model of the inverted pendulum over a cart follows a deterministic MDP for which a controller u can be learned. We take the state space as $\mathbb{R}^4$ and the action space of the force exerted to the cart as $\mathbb{R}$.

The time-discretised formulation of this problem is more common in reinforcement learning settings than in 'classical' control theory. In that case, a a common payoff function is to obtain one unit of reward for each time step that the pendulum is maintained within a threshold of angles. The not linearized, not time-discretized setting, which is more common in optimal control theory, allows the reward, which is usually termed negatively as a cost function J, to have a much more flexible expression, in terms of time spent towards the equilibrium, energy spent to control the device, etc.

$$J(x,a) = \int_0^\infty C(x(t),a(t))\mathrm{d}t$$

2.6 Savings problem example

The *savings problem* is one of the most important models in economics, modelling the dilemma between saving and consumption of resources [29, part IV] (see also [37]). It is also mathematically closely related to the problem of charging a battery, for example choosing when to draw electricity from a power grid to raise the water level in a reservoir [31].

At each discrete time step k, an agent receives an income i_k. They also have a bank balance x_k, which accumulates interest over time (this could also be, for example, an investment portfolio yielding returns). At each time step the agent makes a choice of *consumption*, which means converting their income into utility (or, more literally, things from which they derive utility). If the consumption in some stage is less than their income then the difference is added to the bank balance, and if it is more than the difference is taken from the bank balance. The dilemma is that the agent receives utility only from consumption, but saving gives the possibility of higher consumption later due to interest. The optimal balance between consumption and saving depends on the discount factor, which models the agent's preference between consumption now and consumption in the future.

In the most basic version of the model, all values can be taken as deterministic, and the income i_k can also be taken as constant. This basic model can be expanded in many ways, for example with forecasts and uncertainty about income and interest rates. A straightforward extension, which we will consider in this paper, is that income is normally distributed $i \sim \mathcal{N}(\mu,\sigma)$, independently in each time step.

We take the state space and action space both as $X = A = [0,\infty)$. Given the current bank balance x and consumption decision a, the utility in the current stage is $U(x,a) = \min\{a, x+i\}$. (That is, the agent's consumption is capped by their current bank balance.) The state transition is given by $f(x,a) = \max\{(1+\gamma)x - a + i, 0\}$, where γ is the interest rate.

3 Optics

In this section we recall material on categories of mixed optics, mostly taken from [11].

3.1 Categories of optics

Given a monoidal category $\mathscr{M}$ and a category $\mathscr{C}$, an action of $\mathscr{M}$ on $\mathscr{C}$ is given by a functor $\bullet : \mathscr{M} \times \mathscr{C} \to \mathscr{C}$ with coherence isomorphisms $I \bullet X \cong X$ and $(M \otimes N) \bullet X \cong M \bullet (N \bullet X)$. $\mathscr{C}$ is called an $\mathscr{M}$-actegory.

Given a pair of $\mathscr{M}$-actegories $\mathscr{C}, \mathscr{D}$, we can form the category of optics $\mathbf{Optic}_{\mathscr{C},\mathscr{D}}$. Its objects are pairs $\binom{X}{X'}$ where X is an object of $\mathscr{C}$ and X' is an object of $\mathscr{D}$. Hom-sets are defined by the coend

$$\mathbf{Optic}_{\mathscr{C},\mathscr{D}}\left(\binom{X}{X'}, \binom{Y}{Y'}\right) = \int^{M:\mathscr{M}} \mathscr{C}(X, M \bullet Y) \times \mathscr{D}(M \bullet Y', X')$$

in the category **Set**. Such a morphism is called an optic, and consists of an equivalence class of triples (M, f, f') where M is an object of $\mathscr{M}$, $f : X \to M \bullet Y$ in $\mathscr{C}$ and $g : M \bullet Y' \to X'$ in $\mathscr{D}$. We call M the residual, f the forwards pass and f' the backwards pass, so we think of the residual as mediating communication from the forward pass to the backward pass. Composition of optics works by taking the monoidal product in $\mathscr{M}$ of the residuals.

A common example is a monoidal category $\mathscr{M} = \mathscr{C} = \mathscr{D}$ acting on itself by the monoidal product, so

$$\mathbf{Optic}_{\mathscr{C}}\left(\binom{X}{X'}, \binom{Y}{Y}\right) = \int^{M:\mathscr{C}} \mathscr{C}(X, M \otimes Y) \times \mathscr{C}(M \otimes Y', X')$$

This is the original definition of optics from [33]. If $\mathscr{C}$ is additionally cartesian monoidal then we can eliminate the coend to produce *concrete lenses*:

$$\begin{aligned}\int^{M:\mathscr{C}} \mathscr{C}(X, M \times Y) \times \mathscr{C}(M \times Y', X') &\cong \int^{M:\mathscr{C}} \mathscr{C}(X, M) \times \mathscr{C}(X, Y) \times \mathscr{C}(M \times Y', X') \\ &\cong \mathscr{C}(X, Y) \times \mathscr{C}(X \times Y', X')\end{aligned}$$

On the other hand, if $\mathscr{C}$ is monoidal closed then we can eliminate the coend in a different way to produce *linear lenses*:

$$\begin{aligned}\int^{M:\mathscr{C}} \mathscr{C}(X, M \otimes Y) \times \mathscr{C}(M \otimes Y', X') &\cong \int^{M:\mathscr{C}} \mathscr{C}(X, M \otimes Y) \times \mathscr{C}(M, [Y', X']) \\ &\cong \mathscr{C}(X, [Y', X'] \otimes Y)\end{aligned}$$

Both of these proofs use the *ninja Yoneda lemma* for coends [30].

Example 1. Let **Set** act on itself by cartesian product. Optics $\binom{X}{X'} \to \binom{Y}{Y'}$ in $\mathbf{Optic}_{\mathbf{Set}}$ can be written equivalently as pairs of functions $X \to Y$ and $X \times Y' \to X'$, or as a single function $X \to Y \times (Y' \to X')$.

Example 2. Let **Euc** be the category of Euclidean spaces and smooth functions, which is cartesian but not cartesian closed. Optics $\binom{X}{X'} \to \binom{Y}{Y'}$ in $\mathbf{Optic}_{\mathbf{Euc}}$ can be written as pairs of smooth functions $X \to Y$ and $X \times Y' \to X'$.

Example 3. Let **Mark** be the category of sets and finite support Markov kernels, which is the kleisli category of the finite support probability monad $\Delta : \mathbf{Set} \to \mathbf{Set}$. It is a prototypical example of a Markov category [19], and it is neither cartesian monoidal nor monoidal closed. Optics $\binom{X}{X'} \to \binom{Y}{Y'}$ in $\mathbf{Optic}_{\mathbf{Mark}}$ can only be written as optics, it is not possible to eliminate the coend. This is the setting used for Bayesian open games [6].

Example 4. Let **Conv** be the category of convex sets, which is the Eilenberg-Moore category of the finite support probability monad [18]. A convex set can be thought of a set with an abstract expectation operator $\mathbb{E} : \Delta X \to X$. Thus the functor $\Delta : \mathbf{Mark} \to \mathbf{Conv}$ given by $X \mapsto \Delta(X)$ on objects is fully faithful. **Conv** has finite products which are given by tupling in the usual way. **Conv** also has a closed structure: the set of convex functions $X \to Y$ themselves form a convex set $[X, Y]$ pointwise. However **Conv** is not cartesian closed: instead there is a different monoidal product making it monoidal closed [36, section 2.2] (see also [28]). This monoidal product "classifies biconvex maps" in the same sense that the tensor product of vector spaces classifies bilinear maps. The embedding $\Delta : \mathbf{Mark} \to \mathbf{Conv}$ is strong monoidal for this monoidal product, not for the cartesian product of convex sets.

We can define an action of **Mark** on **Conv**, by $M \bullet X = \Delta(M) \otimes X$ [9, section 5.5]. Together with the self-action of **Mark**, we get a category $\mathbf{Optic}_{\mathbf{Mark},\mathbf{Conv}}$ given by

$$\begin{aligned}\mathbf{Optic}_{\mathbf{Mark},\mathbf{Conv}}\left(\binom{X}{X'},\binom{Y}{Y'}\right) &= \int^{M:\mathbf{Mark}} \mathbf{Mark}(X, M \otimes Y) \times \mathbf{Conv}(\Delta(M) \otimes Y', X') \\ &\cong \int^{M:\mathbf{Mark}} \mathbf{Mark}(X, M \otimes Y) \times \mathbf{Conv}(\Delta(M), [Y', X'])\end{aligned}$$

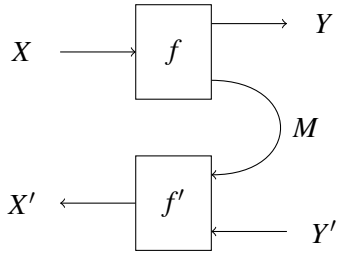
(This coend cannot be eliminated because the embedding $\Delta : \mathbf{Mark} \to \mathbf{Conv}$ does not have a right adjoint.)

This category of optics will be very useful for Markov decision processes, where the forwards direction is a Markov kernel and the backwards direction is a function involving expectations.

3.2 Monoidal structure of optics

A category of optics $\mathbf{Optic}_{\mathscr{C},\mathscr{D}}$ is itself (symmetric) monoidal, when $\mathscr{C}$ and $\mathscr{D}$ are (symmetric) monoidal in a way that is compatible with the actions of $\mathscr{M}$. The details of this have been recently worked out in [9]. The monoidal product on objects of $\mathbf{Optic}_{\mathscr{C},\mathscr{D}}$ is given by pairwise monoidal product. All of the above examples are symmetric monoidal.

A monoidal category of optics comes equipped with a string diagram syntax [24]. This has directed arrows representing the forwards and backwards passes, and right-to-left bending wires but not left-to-right bending wires. The residual of the denoted optic can be read off from a diagram, as the monoidal product of the wire labels of all right-to-left bending wires. For example, a typical optic $(M, f, f') \in \mathbf{Optic}_{\mathscr{C}}\left(\binom{X}{X'},\binom{Y}{Y'}\right)$ is denoted by the diagram



These diagrams have only been properly formalised for a monoidal category acting on itself, so for mixed optics we need to be very careful and are technically being informal.

Costates in monoidal categories of optics, that is optics $\binom{X}{X'} \to I$ (where $I = \binom{I}{I}$ is the monoidal unit of $\mathbf{Optic}_{\mathscr{C},\mathscr{D}}$), are a central theme of this paper. When we have a monoidal category acting on itself, costates in $\mathbf{Optic}_{\mathscr{C}}$ are given by

$$\mathbf{Optic}_{\mathscr{C}}\left(\binom{X}{X'}, I\right) = \int^{M:\mathscr{C}} \mathscr{C}(X, M \otimes I) \times \mathscr{C}(M \otimes I, X') \cong \mathscr{C}(X, X')$$

Thus *costates in optics are functions*. A different way of phrasing this is by defining a functor $K : \mathbf{Optic}_{\mathscr{C}}^{\mathrm{op}} \to \mathbf{Set}$ given on objects by $K\binom{X}{X'} = \mathscr{C}(X, X')$, and then showing that K is representable [25]. We will generally treat this isomorphism as implicit, sometimes referring to costates as though they are functions.

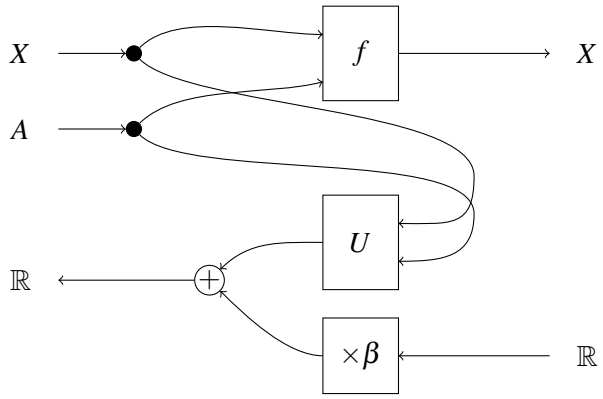
In the case of a cartesian monoidal category $\mathscr{C}$, given a concrete lens $f : X \to Y$, $f' : X \times Y' \to X'$ and a function $k : Y \to Y'$, the action of K gives us the function $X \to X'$ given by $x \mapsto f'(x, k(f(x)))$.

When we have $\mathscr{M} = \mathscr{C}$ acting on both itself and on $\mathscr{D}$ (which includes all of the examples above) then similarly

$$\mathbf{Optic}_{\mathscr{C},D}\left(\binom{X}{X'},I\right) = \int^{M:\mathscr{C}} \mathscr{C}(X, M \otimes I) \times \mathscr{D}(M \bullet I, X') \cong \mathscr{D}(X \bullet I, X')$$

4 Dynamic programming with optics

Given an MDP with state space X and action space A, we can convert it to an optic $\binom{X\otimes A}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$. The category of optics in which this lives can be 'customised' to some extent, and depends on the class of MDPs that we are considering and how much typing information we choose to include. The definition of this optic is given by the following string diagram:

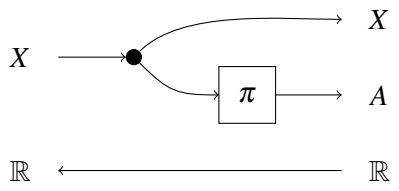


To be clear, this diagram is not completely formal because we are making some assumptions about the category of optics we work in. In general, we require the forwards category $\mathscr{C}$ to be a Markov category (giving us copy morphisms Δ_X and Δ_A), and the backwards category $\mathscr{D}$ must have a suitable object $\mathbb{R}$ together with morphisms $\times\beta : \mathbb{R} \to \mathbb{R}$ and $+ : \mathbb{R} \otimes \mathbb{R} \to \mathbb{R}$. Specific examples of interpretations of this diagram will be explored below. When the forwards category acts on the backwards category, then the forwards pass is a morphism $g : X \otimes A \to X \otimes X \otimes A$ in $\mathscr{C}$ where

$$g = \Delta_{X\otimes A} \fatsemi (f \otimes \mathrm{id}_{X\otimes A})$$

and the backwards pass is a morphism $g' : X \bullet A \bullet \mathbb{R} \to \mathbb{R}$ in $\mathscr{D}$ encoding the function $g'(x,a,r) = \mathbb{E}U(x,a) + \beta r$. The resulting optic is given by $\lambda = (X \otimes A, g, g') : \binom{X\otimes A}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$ in $\mathbf{Optic}_{\mathscr{C},\mathscr{D}}$.

Given a policy $\pi : X \to A$, we lift it to an optic $\overline{\pi} : \binom{X}{\mathbb{R}} \to \binom{X\otimes A}{\mathbb{R}}$, by



Here we are also assuming that the forwards category has a copy morphisms Δ_X (for example, because it is a Markov category), and the backwards category has a suitable object of real numbers. The interpretation of this diagram is the optic $(I, \Delta_X \fatsemi (\mathrm{id}_X \otimes \pi), \mathrm{id}_{\mathbb{R}})$.

4.1 Discrete-space deterministic decision processes

Consider a deterministic decision process with a discrete set of states X, discrete and finite set of actions A, transition function $f : X \times A \to X$, payoff function $U : X \times A \to \mathbb{R}$ and discount factor $\beta \in (0,1)$. We convert this into an optic $\lambda = (X \times A, g, g') : \binom{X \times A}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$ in $\mathbf{Optic}_{\mathbf{Set}}$, whose forwards pass is $g(x,a) = (x,a,f(x,a))$ and whose backwards pass is $g'(x,a,r) = U(x,a) + \beta r$.

Consider a dynamical system with the state space $A^X \times \mathbf{Optic}_{\mathbf{Set}}\left(\binom{X}{\mathbb{R}}, I\right)$. Elements of this are pairs (π, V) of a policy $\pi : X \to A$ and a value function $V : X \to \mathbb{R}$. We can define two update steps:

$$\begin{aligned} &\textbf{Value improvement:} && (\pi, V) \mapsto (\pi, \overline{\pi} \fatsemi \lambda \fatsemi V) \\ &\textbf{Policy improvement:} && (\pi, V) \mapsto (x \mapsto \arg\max_{a \in A} (\lambda \fatsemi V)(x,a), V) \end{aligned}$$

(We assume that arg max is canonically defined, for example because A is equipped with an enumeration so that we can always choose the first maximiser.)

Unpacking and applying the isomorphism between costates in lenses and functions, a step of value improvement replaces V with

$$V'(x) = U(x, \pi(x)) + \beta V(f(x, \pi(x)))$$

and a step of policy improvement replaces π with

$$\pi'(x) = \arg\max_{a \in A} U(x,a) + \beta V(f(x,a))$$

Iterating the value improvement step converges to a value function which is the optimal value function for the current (not necessarily optimal) policy π. A fixpoint of alternating steps of value improvement and policy improvement is a pair (π^*, V^*) satisfying

$$\begin{aligned} V^*(x) &= \max_{a \in A} (\lambda \fatsemi V^*)(x,a) = \max_{a \in A} U(x,a) + \beta V^*(f(x,a)) \\ \pi^*(x) &= \arg\max_{a \in A} (\lambda \fatsemi V^*)(x,a) = \arg\max_{a \in A} U(x,a) + \beta V^*(f(x,a)) \end{aligned}$$

Example 5 (Gridworld example)**.** A policy of an agent in our version of Gridworld (Figure 1) is a function from the 4×4 set of states $X = \{1,2,3,4\}^2$ that we index by (i,j) to the four-element set of actions $A = \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$, i.e. an element of A^X. Initializing the value function V with the environments immediate reward whose only non-zero value is $V(0,0) = 1$ (top-left corner) and the policy with a upwards facing constant action $\pi(i,j) = \uparrow$ for all $(i,j) \in X$, a value improvement step would leave the policy unchanged while updating V to $\overline{\pi} \fatsemi \lambda \fatsemi V$, which differs with V only at $(0,1) \mapsto \beta$.

If we instead perform a policy improvement step, the value function remains unchanged while the new policy differs with π at $(1,0) \mapsto \arg\max_{a \in A} (\lambda \fatsemi v)(1,0,a) = \leftarrow$.

4.2 Continuous-space deterministic decision processes

Example 6 (Inverted pendulum)**.** A state of our time-discretized inverted pendulum on a cart consists of $[y, \dot{y}, \theta, \dot{\theta}]^\top$ in the state space $X = \mathbb{R}^4$. The linearized transition function that sends x_k to $x_{k+1} = Ax_k + Ba_k$ is a smooth map $X \to Y$. The discretized cost $J(x,a) = \sum_{k=0}^{\infty} \beta^k C(x(k), a(k))$ defines the backwards smooth function $X \times A \times \mathbb{R} \to \mathbb{R}$ which adds the cost at the kth time step $C(x(k), a(k))$ to the discounted sum:

$$\left(x(k), a(k), \sum_{j=k+1} \beta^j C(x(j), a(j))\right) \mapsto \sum_{j=k} \beta^j C(x(j), a(j))$$

These two maps form an optic $\lambda : \binom{X\times A}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$ in $\mathbf{Optic}_{\mathbf{Euc}}$. Note that the cost function C is itself typically not affine, but rather convex (intuitively, since the 'good states' that should minimise the cost fall in the middle of the state space).

In conclusion, the two optics involved in this example are

$$\lambda = \begin{pmatrix} f \\ J \end{pmatrix} : \begin{pmatrix} X\times A \\ \mathbb{R} \end{pmatrix} \to \begin{pmatrix} X \\ \mathbb{R} \end{pmatrix} \qquad \overline{\pi} = \begin{pmatrix} \pi \\ p_2 \end{pmatrix} : \begin{pmatrix} X \\ \mathbb{R} \end{pmatrix} \to \begin{pmatrix} X\times A \\ \mathbb{R} \end{pmatrix}$$

$$\begin{aligned} f : X\times A &\to X & \mathrm{gr}(\pi) : X &\to X\times A \\ (x,a) &\mapsto Ax + Ba & x &\mapsto (x,\pi(x)) \\ J : X\times A\times\mathbb{R} &\to \mathbb{R} & p_2 : X\times\mathbb{R} &\to \mathbb{R} \\ (x,a,r) &\mapsto C(x,a) + \beta r & (\_,r) &\mapsto r \end{aligned}$$

This formalisation of the continuous state space misses however a practical problem. Let S be a continuous state space. In numerical implementations, policy improvement over S needs to map an action to every point in the space. Two common approaches are to discretize the state space into a possibly non-uniform grid, or to restrict the space of values to a family of parametrized functions [34, Sec.4.]. The discretization approach treats the continuous state space as a distribution over a simplicial complex X obtained e.g. by triangulation, $\mathbf{Euc}(1,S) \approx \mathbf{Mark}(I,X)$, where a continuous state gets mapped to a distribution over the barycentric coordinates of the simplex. This effectively transforms the initial continuous-space deterministic decision process into a discrete-space MDP, modelling numerical approximation errors as stochastic uncertainty.

4.3 Discrete-space Markov decision processes

Consider a Markov decision process with a discrete set of states X, discrete and finite set of actions A, a transition Markov kernel $f : X\times A \to \Delta(X)$, expected payoff function $U : X\times A \to \mathbb{R}$ and discount factor $\beta \in (0,1)$. We can write the transition function as conditional probabilities $f(x' \mid x,a)$.

We can convert this data into an optic $\lambda : \binom{X\otimes A}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$ in the category $\mathbf{Optic}_{\mathbf{Mark},\mathbf{Conv}}$ given by **Mark** acting on both itself and **Conv**. This optic is given concretely by $(X\otimes A, g, g')$ where $g : X\otimes A \to X\otimes A\otimes X$ in **Mark** is given by $\Delta_{X\otimes A} \fatsemi (f\otimes \mathrm{id}_{X\otimes A})$, and $g' : \Delta(X\otimes A) \to [\mathbb{R},\mathbb{R}]$ in **Conv** is defined by $g'(\alpha)(r) = \mathbb{E}U(\alpha) + \beta r$, where $\alpha \in \Delta(X\times A)$ is a joint distribution on states and actions. Alternatively, we can note that the domain of g' is free on the set $X\times A$ (although it cannot be considered free on an object of **Mark**), and define it as the linear extension of $g'(x,a)(r) = U(x,a) + \beta r$.

With this setup, value improvement $(\pi, V) \mapsto (\pi, \overline{\pi} \fatsemi \lambda \fatsemi V)$ yields the value function

$$V'(x) = \mathbb{E}_{a\sim\pi(x)}[U(x,a) + \beta V(f(x,a))]$$

Alternating steps of value and policy improvement converge to the optimal policy π^* and value function V^*, which maximises the expected value of the policy:

$$V^*_{\pi^*}(x_0) = \mathbb{E}\sum_{k=0}^{\infty} \beta^k U(x_k, \pi^*(x_k))$$

Example 7 (Gridworld, continued)**.** In a proper MDP, transition functions can be stochastic, and update steps have to take expectations over values: value improvement maps $(\pi,V) \mapsto (\pi, \overline{\pi} \fatsemi \lambda \fatsemi V)$ and policy improvement maps $(\pi,V) \mapsto (x \mapsto \arg\max_{a\in A} \mathbb{E}(\lambda \fatsemi V)(x,a), V)$. This model also accepts stochastic policy improvement steps like ε-greedy, which is an ad hoc heuristic technique of balancing exploration and exploitation in reinforcement learning [27, Sec.2], a problem which is known in control theory as the identification-control conflict.

4.4 Continuous-space Markov decision processes

For continuous-space MDPs we need a category of continuous Markov kernels. There are several possibilities for this arising as the kleisli category of a monad, such as the Giry monad on measurable spaces [22], the Radon monad on compact Hausdorff spaces [39] and the Kantorovich monad on complete metric spaces [20]. However, control theorists typically work with more specific parametrised families of distributions for computational reasons, the most common being normal distributions. We will work with the category **Gauss** of Euclidean spaces and affine functions with Gaussian noise [19, section 6]. (This is an example of a Markov category that does not arise as the kleisli category of a monad, because its multiplication map would not be affine.) This works because the pushforward measure of a Gaussian distribution along an affine function is still Gaussian, which fails for more general functions.

Example 8. We will formulate the savings problem with normally-distributed income. The inequality constraints (namely that the balance cannot be negative and that the agent cannot consume more than their current balance) introduce nonlinearities. We can deal with the latter by constraining the optimisation in the policy improvement step, but the former threatens to take us outside the category **Gauss** and we must allow the balance to possibly become negative for the purposes of this example.

Gauss is a Markov category that is not cartesian (the monoidal product is the cartesian product of Euclidean spaces, which adds the dimensions), so it acts on itself by the monoidal product and we take the category $\mathbf{Optic}_{\mathbf{Gauss}}$. We take the state and action spaces to be $X = A = \mathbb{R}$. The transition function $f : \mathbb{R} \otimes \mathbb{R} \to \mathbb{R}$ is given by $f(x,a) = (1+\gamma)x - a + \mathcal{N}(\mu, \sigma)$, and the payoff function $U : \mathbb{R} \otimes \mathbb{R} \to \mathbb{R}$ is given by $U(x,a) = a$.

We modify the policy improvement step to be

$$\textbf{Policy improvement:} \qquad (\pi, V) \mapsto (x \mapsto \arg\max_{a \in A(x)} (\lambda \fatsemi V)(x,a), V)$$

where $A(x)$ is the set $A(x) = \{a \in \mathbb{R} \mid 0 \leq a \leq x + i\}$. This enforces that the agent cannot consume negative amounts or consume more than their current balance - since the optimisation is done externally to the category **Gauss** we can avoid one source of nonlinearity this way.

5 Q-learning

Consider a deterministic decision process corresponding to the optic $\lambda : \binom{X \times A}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$. The dynamical system with state space $A^X \times \mathbf{Optic}_{\mathbf{Set}}\left(\binom{X \times A}{\mathbb{R}}, I\right)$ has elements (π, q) consisting of a *state-action* value function $q : X \times A \to \mathbb{R}$ as in (4) rather than a state-value function $V : X \to \mathbb{R}$.

We can define similar update steps

$$\textbf{Value improvement:} \qquad (\pi, q) \mapsto (\pi, \lambda \fatsemi \overline{\pi} \fatsemi q)$$

$$\textbf{Policy improvement:} \qquad (\pi, q) \mapsto \left(x \mapsto \arg\max_{a \in A} q(x,a), q\right)$$

These can also be fused into a single step:

$$\textbf{State-action value iteration:} \qquad (\pi, q) \mapsto \left(x \mapsto \arg\max_{a \in A} q(x,a), \lambda \fatsemi \overline{\pi} \fatsemi q\right)$$

Observe that composition of the λ optic with $\overline{\pi}$ is flipped compared to the case seen in Section 4.1, as we want an element of $\mathbf{Optic}_{\mathbf{Set}}\left(\binom{X \times A}{\mathbb{R}}, \binom{X \times A}{\mathbb{R}}\right)$ to compose with q.

The advantage of learning state-action value functions $X \otimes A \to \mathbb{R}$ rather than state-value functions $X \to \mathbb{R}$ is that is gives a way to approximate $\arg\max_{a \in A}(\lambda \fatsemi \overline{\pi} \fatsemi q)(x,a)$ without making any use of λ, namely by instead using $\arg\max_{a \in A} q(x,a)$. This leads to an effective method known as Q-learning for computing optimal control policies even when the MDP is unknown, with only a single state transition and payoff being sampled at each time-step. This is the essential difference between classical control theory and *reinforcement learning*. The above method, despite learning a q-function, is *not* Q-learning because it makes use of λ during value improvement.

Q-learning [23] is a sampling algorithm that approximates the state-action value iteration, usually by a lookup table Q referred as Q-matrix. It treats the optic as a black box, having therefore no access to the transition or rewards functions used in (4), and instead updates q by interacting with the environment dynamics:

$$q'(x',a) = (1-\alpha)q(x,a) + \alpha(r + \beta \max_{a'} q(x',a'))$$

where $\alpha \in (0,1)$ is a weighting parameter. Note that both the new state x' and the reward r are obtained by interacting with the system, rather than looked ahead by $x' = f(x,a)$ and $r = U(x,a)$. It falls in the family of temporal difference algorithms.

6 Further work

At the end of the previous section, it can be seen that Q-learning is no longer essentially using the structure of the category of optics, instead treating the Q-function as a mere function. We believe this can be overcome using the framework of categorical cybernetics [14], leading to a fully optic-based approach to reinforcement learning. By combining with other instantiations of the same framework, it is hoped to encompass the zoo of modern variants of reinforcement learning that have achieved spectacular success in many applications in the last few years. For example, deep Q-learning represents the Q-function not as a matrix but as a deep neural network, trained by gradient descent, allowing much higher dimensionality to be handled in practice. Deep learning is currently one of the main applications of categorical cybernetics [12].

The proof that dynamic programming algorithms converge to the optimal policy and value function typically proceed by noting that the set of all value functions form a complete ordered metric space and that value improvement is a monotone contraction mapping. The metric structure is used to prove that iteration converges to a unique fixpoint by the contraction mapping theorem, and then the order structure is used to prove that this fixpoint is indeed optimal. Since value improvement is optic composition, these facts would be a special case of the category of optics being enriched in the category of ordered metric spaces and monotone contraction mappings. We do not currently know whether such an enrichment is possible. Unlike costates, general optics have nontrivial forwards passes, so there are two possible approaches: either ignore the forwards passes and defining a metric only in terms of the backwards passes, or defining a metric also using the forwards passes, for example using the Kantorovich metric between distributions. This would also be a reasonable place to unify our approach with the coalgebraic approach with metric coinduction [16].

Finally, continuous time MDPs pose a serious challenge to any approach for which categorical composition is sequencing in time, since composition of two morphisms in a category appears to be inherently discrete-time. (Open games are similarly unable to handle dynamic games with continuous time, such as pursuit games.) A plausible approach to this is to associate an endomorphism in a category to every real interval, by treating that interval of time as a single discrete time-step, and then requiring that all

morphisms compose together correctly, similar to a sheaf condition. It is hoped that the Bellman-Jacobi-Hamilton equation, a PDE that is the continuous time analogue of the discrete-time Bellman equation, will simiarly arise as a fixpoint in this way. Exploring this systematically is important future work.

References

[1] K.J Åström (1965): *Optimal control of Markov processes with incomplete state information*. Journal of Mathematical Analysis and Applications 10(1), pp. 174–205, doi:10.1016/0022-247x(65)90154-x.

[2] John Baez & Jason Erbele (2015): *Categories in control*. ArXiv:1405.6881.

[3] Wolfram Barfuss (2019): *Learning dynamics and decision paradigms in social-ecological dilemmas*. Ph.D. thesis, Humboldt-Universität zu Berlin.

[4] Richard Bellman (1957): *Dynamic programming*. Princeton University Press.

[5] Dimitri P. Bertsekas (2007): *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific.

[6] Joe Bolt, Jules Hedges & Philipp Zahn (2019): *Bayesian open games*. Forthcoming in *Compositionality*, arXiv:1910.03656.

[7] Nicola Botta, Cezar Ionescu & Edwin Brady (2013): *Sequential decision problems, dependently typed solutions*. In: Proeedings of PLMMS13.

[8] Dylan Braithwaite, Matteo Capucci, Bruno Gavranović, Jules Hedges & Eigil Fjeldgren Rischel (2022): *Fibre optics*. ArXiv:2112.11145.

[9] Matteo Capucci & Bruno Gavranović (2022): *Actegories for the working amthematician*. ArXiv:2203.16351.

[10] Matteo Capucci, Neil Ghani, Jérémy Ledent & Fredrik Nordvall Forsberg (2021): *Translating extensive form games to open games with agency*. Forthcoming in *Proceedings of ACT 2021*.

[11] Bryce Clarke, Derek Elkins, Jeremy Gibbons, Fosco Loregian, Bartosz Milewski, Emily Pillmore & Mario Román (2020): *Profunctor optics: A categorical update*. ArXiv:2001.07488.

[12] G.S.H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson & Fabio Zanasi (2021): *Categorical foundations of gradient-based learning*. ArXiv:2103.01931.

[13] Eric V. Denardo (1967): *Contraction Mappings in the Theory Underlying Dynamic Programming*. SIAM Review 9(2), pp. 165–177, doi:10.1137/1009030. Available at `https://ui.adsabs.harvard.edu/abs/1967SIAMR...9..165D`.

[14] Matteo Capucci Bruno Gavranović Jules Hedges Eigil Fjeldgren Rischel (2021): *Towards foundations of categorical cybernetics*. Forthcoming in *Proceedings of ACT 2021*, arXiv:2105.06332.

[15] Nicolas Eschenbaum, Filip Mellgren & Philipp Zahn (2022): *Robust algorithmic collusion*. ArXiv:2201.00345.

[16] Frank M. V. Feys, Helle Hvid Hansen & Lawrence S. Moss (2018): *Long-Term Values in Markov Decision Processes, (Co)Algebraically*. In Corina Cîrstea, editor: Coalgebraic Methods in Computer Science, 11202, Springer International Publishing, Cham, pp. 78–99. Available at `http://link.springer.com/10.1007/978-3-030-00389-0_6`. Series Title: Lecture Notes in Computer Science.

[17] Bernard Friedland (1985): *Control Systems Design*. McGraw-Hill Companies.

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

[19] 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.

[20] Tobias Fritz & Paolo Perrone (2019): *A probability monad as the colimit of spaces of finite samples*. Theory and applications of categories 34(7).

[21] Neil Ghani, Jules Hedges, Viktor Winschel & Philipp Zahn (2018): *Compositional game theory*. In: *Proceedings of Logic in Computer Science (LiCS) 2018*, ACM, pp. 472–481, doi:10.1145/3209108.3209165.

[22] Michèlle Giry (1982): *A categorical approach to probability theory*. *Categorical aspects of topology and analysis*, pp. 68–85, doi:10.1007/BFb0092872.

[23] Watkins Christopher J. C. H. & Dayan Peter (1992): *Q-learning*. *Machine Learning*, doi:10.1.1.466.7149.

[24] Jules Hedges (2017): *Coherence for lenses and open games*. ArXiv:1704.02230.

[25] Jules Hedges (2018): *Morphisms of open games*. In: *Proceedings of MFPS 2018*, *Electronic notes in theoretical computer science* 341, pp. 151–177.

[26] David Jaz Myers (2021): *Double Categories of Open Dynamical Systems (Extended Abstract)*. *Electronic Proceedings in Theoretical Computer Science* 333, pp. 154–167, doi:10.4204/EPTCS.333.11. Available at `http://arxiv.org/abs/2005.05956v2`.

[27] 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.

[28] Anders Kock (1971): *Closed categories generated by commutative monads*. *Journal of the Australian Mathematical Society* 12(4).

[29] Lars Ljungqvist & Thomas Sargent (2004): *Recursive macroeconomic theory*. MIT Press.

[30] Fosco Loregian (2021): *Coend calculus*. Cambridge University Press.

[31] Sagar Mody & Thomas Steffen (2015): *Optimal charging of electric vehicles using a stochastic dynamic programming model and price prediction*. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems* 8(2).

[32] Martin L. Puterman (2005): *Markov decision processes: discrete stochastic dynamic programming*. Wiley series in probability and statistics, Wiley-Interscience, Hoboken, NJ.

[33] Mitchell Riley (2018): *Categories of Optics*. *arXiv:1809.00738 [math]*. Available at `http://arxiv.org/abs/1809.00738`. ArXiv: 1809.00738.

[34] John Rust (1996): *Numerical dynamic programming in economics*. In: *Handbook of computational economics. Vol. 1*, Amsterdam: Elsevier, pp. 619–729.

[35] David I. Spivak (2020): *Generalized Lens Categories via functors C^op->Cat*. *arXiv:1908.02202 [cs, math]*. Available at `http://arxiv.org/abs/1908.02202`. ArXiv: 1908.02202.

[36] Kirk Stirtz (2015): *Categorical probability theory*. ArXiv:1406.6030.

[37] Nancy Stokey, Robert Lucas & Edward Prescott (1989): *Recursive methods in economic dynamics*. Harvard University Press.

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

[39] T. Świrszcz (1974): *Monadic functors and convexity*. *Bulletin de l'Acaédemie Polonaise des Sciences* 22(1).

[40] Pietro Vertechi (2022): *Dependent optics*. ArXiv:2204.09547.