Categorical Models in Control Theory & Reinforcement Learning

PhD Thesis

Riu Rodríguez Sakamoto

Mathematically Structured Programming Group

Computer and Information Sciences

University of Strathclyde, Glasgow

October 27, 2025

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Acknowledgements

Deepest appreciation goes to my supervisor and mentor Jules Hedges for showing me what 'applied' in '(applied) applied category theory' means. I am very thankful to my second supervisor Neil Ghani as well, for the little nuggets of wisdom that throw me into bottomless rabbit holes and the piercing advice on communicating ideas.

I wish to extend my sincerest gratitude to the Applied Category Theory community, for their brilliance, open arms and the sense of belonging they fostered. The brief but inspiring conversations with David Jaz Myers, David Spivak, Fabrizio Genovese, Nima Motamed, and others that I had during my first attendance in 2022 greatly influenced the direction of my research in the years that followed. This research could also not have been possible without the academic environment and shared space provided by the Mathematically Structured Programming group. I thank all members with whom I shared chats and pints and made me raise my head from being stuck in a non-problem, including Bob, Clemens, Conor, Fred, Guillaume, Alasdair, Malin, Sean, Georgi, Ezra, Joe, Dilsat, Sam, Aven, any many more who came by LT1310. Also to the 'cybernetics'-adjacent group of mostly Strathclyde-based friends and colleagues including Matteo Capucci, Bruno Gavranović, Toby Smithe, André Videla, Dylan Braithwaite, Zanzi, Jade Master, Eigil Rishel, and others with whom I had many stimulating conversations and banter. Thanks to davidad, the organisers of CATNIP and other conferences and workshops that I had the chance to be part of, for providing venues of collaboration and their pivotal role in connecting researchers. My heartfelt thanks also go to Bruce Stephen and the ENSIGN group, for supporting me on the last stretch of my doctoral studies and giving me the opportunity to work on 'real world' problems. I thank all the often unseen hands that make everything work without people noticing.

To my parents, who endured and supported from afar and listened to endless rants in moments of despair. To Andrea, who has stuck by me in these particularly defining periods of our journeys.

Contents

Contents

Chapter 1

Introduction

In the scientific inquiry, the integration of disparate fields from innocent connections often yields new perspectives. This thesis explores the bidirectional nature of some methods in Control Theory and Reinforcement Learning. The languages in which these problems are stated are varied as they seek to address different questions, while engaging with common structures. Our aim is to illuminate some bridges between these applied fields, and we believe that category theory is the right language for this.

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, including topological data analysis [72]) and *variational learning* (Bayesian inference and related probabilistic methods). It can be seen as an extension of *dynamic programming* methods in optimal control theory [24], 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 [146] defeating the human Go champion in 2016 and AlphaStar [164] achieving Grandmaster status in the real time strategy game StarCraft II.

Category theory builds the foundations of several mathematical fields, and one of its defining ideas is that of compositionality—the ability to build complex systems systematically from simpler components—which also permeates the applied sciences

where modularity is sought out, e.g. by allowing individual parts of a learning or control system to be analysed or replaced without disrupting the whole. Such a categorical perspective not only clarifies theoretical relationships between models but also guides the effective design, analysis, and implementation of scalable, interpretable systems. From this point of view there has been work both in the classic and modern literature [101, 65]. Categorical Systems Theory [125, 128] studies *systems* as first class objects. The characterisation of open systems by Fong et.al. [59] builds on the classical control literature [140] and Willems' study of behaviours of dynamical systems [167] to advocate linear time-invariant systems over Kalman's analysis of state-space models [66]:

> It is remarkable that the idea of viewing a system in terms of inputs and outputs, in terms of cause and effect, kept its central place in systems and control theory throughout the 20th century. Input/output thinking is not an appropriate starting point in a field that has modeling of physical systems as one of its main concerns. — J.C.Willems [167]

During the author's doctoral studies, being introduced to the field of Applied Category Theory (ACT) through the theory of Open Games [82], now evolved to Cybernetic Systems [42], the task of surmising a connection between these topics which respected enough details about simple examples in Control Theory came with a lot of uncertainty. However from the practitioners point of view, the attention to these details is of utmost importance, and abstracting them away without 'going back down the ladder' [162] and characterising some examples that motivate their study in the first place may be deemed not useful. Thus a common ground found in chapter 4 of this thesis is the study of time-*variant* systems, revisiting the state-space approach by Kalman.

1.1 Relational vs directional theories

ACT may be thought of as the usage of category theoretic gadgets as a tool to describe certain aspects of existing or new methods in fields such as control theory or machine learning, as close as possible to the honest problem-solving motivations that drive their research. In the practice of ACT, one has to balance between the *problem description*

and the *solution methodology*, and this gap depends on the problem at hand. Problems characterised by a narrower gap tend to preserve more structure from domain knowledge, while for others this gap is broader. While instances with a smaller gap are often perceived as more 'elegant', this frequently overlooks that the problem description itself may begin already at an already abstracted level.

Certain categorical gadgets, like the theory of hypergraphs [105, 62], decorated cospans [59, 11, 12, 13], and PROPs [116, 109, 32] are particularly well-suited to the study of network-like, relational theories. We call them 'relational theories' informally to give the intuition of structures that lack inherent directionality, akin to the category **Rel** of sets and relations. These find their application e.g. in the analysis of structural properties of graphs, and are well-suited for what we referred to as *problem descriptions* at the beginning of this section.

Conversely, other algorithms in applied fields have choices that are justified not from the algebraic description of the problem, but from efficiency or other practical considerations in the computation of a solution. Instead of focusing on the structure analysis for example, they are primarily concerned with traversal strategies, and they have a set-like flavour. Rather than considering these *solution methods* as *ad hoc*, we will consider them in the language of 'directional' or 'process' theories. Theories of lenses, optics [136, 132] and open games [82, 71] are some examples.

In some cases, the steady state behaviour of these systems can be functorially mapped back to a network theory via *black-boxing* functors [60], at the expense of losing transient regime behaviour.

1.2 Overview and contributions

Chapter 2 provides a review of the main category theory results used throughout the rest of the thesis. The main original contributions, summarised in the table below, are mostly around the category of decorated spans.

String diagrams for continuations of optics in the free autonomisation of **Set**	Figure 2.4
Definition of decorated spans (heavily inspired by decorated cospans [59])	Definition 2.6.5
Relation between weighted para and decorated spans	Proposition 2.6.8

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. One of its fundamental constructs is the Bellman operator [17]. In chapter 3 we analyse its most basic formulation in the context of Markov decision processes (MDPs), and we capture its bidirectionality in the framework of lenses and optics. From an earlier version of this chapter written jointly with J. Hedges [85], some of the original results are shown below.

Proof that policy improvement is not an optic	Proposition 3.4.2
Identification of several Markov decision processes in the language of optics	Section 3.5

In control theory, the Bellman operator appears in the problem of computing optimal controls for linear quadratic regulators (LQRs). Chapter 4 is an exercise in characterising a solution method that is in any way more efficient than exhaustive search through the elements of a set and exploring how much structure is needed to do so. It exploits the algebraic structure of quadratic value functions and linear dynamics to characterise what could be called the incarnation of Bellman operators in linear algebra, the Riccati equation, functorially in the dynamics. This requires a combination of the problem descriptions and solution methods discussed above, by building a directional theory on top of a rich enough relational theory.

The aim is to provide a constructive answer to an engineering problem: how is an optimal control actually designed? This practical focus leads to an approach that may lack elegance from a category-theoretic perspective, highlighting a significant challenge encountered in interdisciplinary research. However, the more detailed approach pays off in several ways. Theorem 4.4.3 proves that Riccati difference equations are functorial,

for dynamics that are not necessarily time-invariant, that is, not endomorphic. This is both grounded in the standard linear algebra approach to these optimisation problems, and gives a category-theoretic language in which the potential connections with other fields are made structurally apparent. The entire chapter is original contribution. We highlight the main results below.

Identification of the Löwner order as the adequate order between symmetric matrices to talk about 'optimal choices'	Remark 4.2.6
Definition of the indexed category of symmetric matrices $\mathrm{Sym}_\bullet : \mathbf{FVec} \to \mathbf{Cat}$	Proposition 4.2.10
Description of the delayed rewards problem as an efficient operation in a category of F-lenses	Section 4.2.1
Proof of pointed gaussian distributions being fibred over affine maps	Proposition 4.2.26
Specification of the linear quadratic regulator (LQR) problem as morphisms in a certain category, and functoriality of the solution	Theorem 4.4.3
Specification of the extended Kalman filter (EKF) problem as morphisms in a certain category and functoriality of the solution	Theorem 4.4.5
Specialisation of the time-invariant cases of LQR and EKF as endomorphisms in such categories	Section 4.4.3
Duality between categories of biparametrised morphisms, as an extension of the classical $\mathrm{Hom}(-, k)$ duality of $\mathbf{FVec}_k$	Definition 4.4.11
Control-estimation duality as a functor realising the bijection between linear quadratic regulators and (extended) Kalman filters	Theorem 4.4.13

Having studied the Bellman equation in an environment with a lot of structure, chapter 5 focuses on the other direction: dealing with an unknown environment. Reinforcement Learning (RL) comprises a diverse set of methods that address fundamental challenges in sequential decision-making that arise from interacting with an environment whose structure is not known at all. These methods do not seek a logically extensional nor intensional solution concept: there's no list of 'correct weights' of a model, nor a function of the weights that asserts whether a trained model is correct

or not. Instead, they are grounded in both experimental insights and formal analyses, aiming to overcome persistent issues such as the credit assignment problem, the exploration-exploitation dilemma, and the difficulties posed by partially observable or high-dimensional state spaces. Notably, these challenges were also recognized in earlier disciplines, including psychology and neuroscience [98].

Because of these assessment challenges, most of the mathematical modelling in chapter 5 cannot assume much more than the category **Set**, and characterises concepts in RL like agent, environments and models in the framework of parametrised optics. A central piece of this chapter is the concept of feedback from *samples*, by which RL algorithms learn. The notion of feedback in MDPs has been captured as metric contractions many times, and the existence of a fixpoint for MDPs too [108]. The aim of this chapter is however to focus on what R. Sutton calls 'evaluative feedback' [158, Ch.2], by which RL algorithms operate on top of MDPs (like Q-learning) and their respective feedback and contraction results. This is also different from the 'instructive feedback' that appears in other Machine Learning areas and is independent of the action taken, simply indicating which action is the correct one. The topic of RL is still connected to MDPs, as the MDPs are the usual formalisation of the internal model of a RL agent, i.e. the substrate over which it decides which actions to take. The main original contributions—including some that already appeared in the paper [84]—are the following.

Definition of empirical Bellman operators as parametrised optics	Definition 5.4.1
Classification of RL models in terms of their interfaces	Figure 5.5
Classification of RL environments in terms of their interfaces	Figure 5.8

Chapter 2

Category theory

2.1 Categories for the AI researcher

Repetition is tedious. Finding repeating patterns in code, mathematics and elsewhere and giving them names makes the communication of ideas about patterns more efficient. But both parties need to know what those names refer to. One such concept is a graph. It consists of a collection of vertices and a collection of edges, where each edge links a pair of vertices. A category is similar to a graph; it has a collection of *objects* ($\sim$ vertices) and a collection of maps or *morphisms* ($\sim$ edges). Every morphism f has a *domain* or *source* object $s(f)$ and a *codomain* or *target* object $t(f)$. A category has some additional requirements: for every pair of morphisms f and g for which the codomain of the first is the domain of the second ($t(f) = s(g)$) there exists a composite morphism $g \circ f : s(f) \to t(g)$. We also sometimes use the 'diagrammatic' notation $f \fatsemi g$ synonymously to $g \circ f$. Also, for every object x, there is an identity morphism $\mathrm{id}_x : x \to x$. The composites and identities are not arbitrary though; they have to satisfy some axioms. In particular, associativity ($f \fatsemi (g \fatsemi h) = (f \fatsemi g) \fatsemi h$ for morphisms f, g, h that are composable, i.e. when $t(f) = s(g)$ and $t(g) = s(h)$) and unitality ($f \fatsemi \mathrm{id}_{t(f)} = f = \mathrm{id}_{s(f)} \fatsemi f$).

Examples of categories include those of mathematical structures: The category **Set** has sets as objects, and functions between sets as morphisms. We can quickly check that every set has an identity function on it, which sends every element to itself, and

for any pair of functions $f : A \to B$ and $g : B \to C$, $f \fatsemi g : A \to C$ is defined as one would expect. For a field k, the category $\mathbf{Vec}_k$ consists of vector spaces over k and k-linear maps between them. We will discuss finite vector spaces in the next section too.

Much like undirected graphs, morphisms in a category are not always 'directional'; the category **Rel** for example has sets as objects (just like **Set**), but a morphism $r : A \to B$ in **Rel** is a relation between A and B, i.e. a subset of $A \times B$, or alternatively a function $A \times B \to 2$ where 2 is the 2-element set.

If one is overwhelmed with categories whose collection of objects and morphisms are infinite, one may also look into smaller categories; the smallest one being the *empty category*, denoted **0**, has no objects and no morphisms. The *terminal category*, denoted **1**, has one object and its corresponding identity morphism. Categories that contain a set of objects and only identity morphisms are called *discrete*, and both **0** and **1** are examples. The category **2** is another discrete category which consists of two objects and their respective identities. The category $\mathcal{I}$ is a non-discrete category that contains the same objects as **2** and an additional morphism $0 \to 1$. It is often the case that simple constructions like **2**, $\mathcal{I}$ the (non-)discreteness of categories **n** for higher n have lots of different names and slight variations depending on the context, so we will spell out what we mean when using these notations. Coming back to the beginning of this section, any graph $G = (V, E)$ can be canonically turned into a category whose objects are the vertices V and whose morphisms are the edges E and all compositions of them, including identity edges on the vertices. This is called the *free category* on the graph G.

Often non-examples are just as clarifying when learning about a new concept. A graph G on its own does not a priori form a category, as it would be missing identity morphisms and their compositions. Sometimes the restrictions that one may impose on objects or morphisms makes the definition of a category not possible. For example, consider the natural numbers $\mathbb{N}$ as objects of a category. If one tried to define morphisms to only add one, $(+1) : n \to n + 1$, this would fail because $(+1) : n \to n + 1$ and $(+1) : n + 1 \to n + 2$ compose to $(+2) : n \to n + 2$. If one were to relax the definition

of morphisms to be strictly increasing maps $(<) : n \to m$ where $n < m$, compositions would exist, but identity morphisms would not. Therefore, the set $\mathbb{N}$ is usually thought of as a category where morphisms $n \to m$ exist if $n \leq m$. We will see in the next section that there are many more interesting categories whose objects are $\mathbb{N}$.

One of the most important concepts in category theory is that of universal properties. An example of universal property is the characterisation of certain objects in a category.

Definition 2.1.1. An *initial object* in a category $\mathcal{C}$ is an object 0 such that for all objects $x \in \mathcal{C}$, there is a unique morphism $0 \to x$. A *terminal* or *final object* in $\mathcal{C}$ is an object 1 such that for all objects $x \in \mathcal{C}$, there is a unique morphism $x \to 1$. If an object z in $\mathcal{C}$ is both initial and final, it is called the *zero object*.

Looking at the categories introduced so far, the category **Set** has a terminal object 1, namely the set $\{*\}$ with a single element. Indeed, for any set X, there is always a function $f : X \to 1$ that takes every element of X to $*$, and this function is unique. **Set** does also have an initial object 0, the empty set $\emptyset$. There is a unique function $f : 0 \to X$, by vacuous truth. In **Rel**, the empty set is both the initial and terminal object, i.e. the zero object. In $\mathbf{Vec}_k$, the zero-dimensional vector space k^0 is the zero object. The category **Field** of fields as objects and field homomorphisms as morphisms does *not* have initial nor terminal objects. Because having initial and/or terminal objects is a fairly basic requirement for other constructions in category theory, this category is sometimes deemed not to be very well-behaved. The category **2**, which does not contain a morphism $0 \to 1$, has no initial nor terminal object, but $\mathcal{I}$, which has a morphism $0 \to 1$, has initial object 0 and terminal object 1.

A category $\mathcal{C}$ is said to be *monoidal* if it has a functor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ called the *monoidal product*, and an object I called the unit, that satisfy certain properties (see definition A.1.1). For this introduction, monoidal structure formalises the idea of 'parallel composition'. We denote a monoidal category by $(\mathcal{C}, \otimes, I)$.

For every object Y in a monoidal category $(\mathcal{C}, \otimes, I)$ one may construct the 'tensoring with Y' functor $(- \otimes Y) : \mathcal{C} \to \mathcal{C}$ that maps $X \mapsto X \otimes Y$. When this functor has a right adjoint functor, typically denoted $[Y, -] : \mathcal{C} \to \mathcal{C}$ and called 'internal hom', $\mathcal{C}$ is said

to be *closed* under that monoidal product; it is a closed monoidal category. Moreover, if the monoidal product under which a category is closed coincides with the cartesian product, $\mathcal{C}$ is said to be *cartesian closed*.

This is only a cursory, motivational introduction to category theory, reserving thorough treatment to classical books [115, 112], and expositions of the ACT-adjacent theory [63, 125].

2.2 Be free if possible, concrete if not

The set-theoretic basis of a lot of mathematical education, particularly in applied STEM disciplines, makes the *concreteness* approach to category theory incredibly easy for newcomers. The definition of the *concrete* category of finite-dimensional vector spaces $\mathbf{FVec}_k$ over a field k consists of finite k-vector spaces as objects and k-linear maps as morphisms. Objects are thus sets equipped with extra structure.

Treating vectors as elements of a vector space $v \in V$ has some risks such as an implicit choice of basis or the fact that the definition of structure on objects (linearity) has to coincide with the structure preserved by morphisms. We can alternatively characterize a vector by a map to V that points to the vector. The category $\mathbf{FVec}_k$ contains the object k^1, the one-dimensional space, thus a linear map $k^1 \to V$ that maps the unit vector in k^1 to $v \in V$ is in one-to-one correspondence with the element v. So considering elements as linear maps $k^1 \to V$[1], a lot of these issues are solved (by e.g. unifying the structure to be defined only on morphisms) or moved elsewhere (like the choice of basis, which becomes an issue of identifying the unit vector in k^1). This motivates the structuralist definition of $\mathbf{FVec}_k$, where objects are *natural numbers* corresponding to the dimension of the vector space it refers to.

Definition 2.2.1. Let k be a field. We write $\mathbf{FVec}_k$ for the category with objects $n : \mathbb{N}$, and morphisms $n \to m$ being k-linear maps $k^n \to k^m$.

When talking about 'elements of an object' in a category like $\mathbf{FVec}_k$, we understand them to be morphisms from an appropriate universal object. The end of this section

[1] These maps from the terminal object are called *generalized elements*

will clarify what 'like $\mathbf{FVec}_k$' means specifically.

We mainly use the monoidal product on $\mathbf{FVec}_k$ given by the direct sum or equivalently the direct product of finite dimensional vector spaces, denoted $+$ (with unit 0)[2], and write $(\mathbf{FVec}_k, +, 0)$ to refer to the monoidal category. The operator $+$ is called the *biproduct* as it has the universal property of both products and coproducts. This equivalence between products and coproducts is not satisfied by categories of non-finite dimensional spaces. In $\mathbf{Vec}_k$ for example, coproducts (direct sums) are generally smaller than products (direct products): $\bigoplus_{i\in I} V_i$ consists of tuples $(v_i) \in \prod_{i\in I} V_i$ (or rather, morphisms $\sum_{i:I} i \to \prod_{i:I} i$ applying the freeness principle above) such that only finitely many of the v_i are nonzero, while $\prod_{i\in I} V_i$ has no such restrictions. When considering finite-dimensional objects, there's no difference between 'finitely many entries' and 'any entries', and morphisms can always be represented as matrices without invoking the axiom of choice.

Another monoidal product on $\mathbf{FVec}_k$ is the tensor product $\times$ (with unit 1)[3]. We write $\mathbf{FVec}$ without subscript to refer to the category of finite-dimensional *real* vector spaces (i.e. over $k = \mathbb{R}$), although many of the results, especially in chapter 4, are appropriately generalizable to $k = \mathbb{C}$.

In a category $\mathcal{C}$, the 'free approach' using generalized elements recovers the objects X as sets via the isomorphism $\mathcal{C}(I, X) \cong X$ called the *Yoneda isomorphism* [104]. Nevertheless, this only recovers the underlying set of e.g. vector spaces as objects of $\mathbf{FVec}$. Given a closed symmetric monoidal category $\mathcal{V}$, the appropriate (strong) Yoneda lemma applies for $\mathcal{V}$-enriched categories, where the Yoneda isomorphism is witnessed in $\mathcal{V}_0$. We will be interested in categories enriched in themselves. Some examples include $\mathbf{FVec}$, the category $\mathbf{Pos}$ of posets and monotone maps and the category $\mathbf{Mod}_R$ of modules over a commutative ring R. Chapter 4 will make use of the category $\mathbf{SMod}_{\mathbb{R}_+}$ of semimodules over the semiring of positive reals, which is also enriched over itself since it is symmetric monoidal with biproducts [89]. When the base of enrichment of a category is not itself, one can only recover partial algebraic structure of objects from generalized elements.

[2] In concrete terms, $\oplus$ with unit k^0
[3] In concrete terms, $\otimes$ with unit k.

Definition 2.2.2. The category $\mathbf{Mfd}_{\geq k}$ consists of k-differentiable manifolds (with charts on finite euclidean spaces) as objects and k-differentiable functions between them as morphisms. The product of two objects M, N is given by the cartesian product of their underlying sets $M \times N$, equipped with the product topology and smooth structure given by product charts on $\mathbb{R}^{\dim M + \dim N}$.

The use of differential geometry in this work is minimal, limited to concepts such as manifolds, tangent bundles and vector fields used to illustrate the inverted pendulum example (section 3.3.3) and the Bellman operator (section 5.4). Readers seeking a more detailed mathematical background may consult [110].

The category $\mathbf{Mfd}_\infty$ (or simply $\mathbf{Mfd}$) of smooth manifolds is not cartesian closed, as the space $\mathcal{C}^\infty(M, N)$ of smooth maps is not a smooth manifold. Consider e.g. $M = \mathbb{R}$ and $M = \mathbb{R}$. $\mathcal{C}^\infty(\mathbb{R}, \mathbb{R})$ is infinite-dimensional, thus cannot be modelled locally on $\mathbb{R}^n$.

A 'free' definition of a subcategory of $\mathbf{Mfd}_\infty$ can be constructed for manifolds that are globally diffeomorphic to $\mathbb{R}^n$ for some n. We call this category **Smooth**.

Definition 2.2.3. The category **Smooth** of euclidean spaces—with their topology induced from the euclidean metric—and smooth functions consists of objects $n : \mathbb{N}$ and morphisms $n \to m$ being smooth maps $\mathbb{R}^n \to \mathbb{R}^m$.

We write $\mathbf{Smooth}_\bullet$ for the category of pointed euclidean spaces and point-preserving smooth maps, and generally use the subscript $(-)_\bullet$ to refer to pointed categories[4].

2.3 Generalized lenses, charts and the Grothendieck construction

The study of lens combinators was partly motivated by the problem of bidirectional (view-update) transformations on datastructures [64]. *Lenses* are pairs of a *view* map $v : X \to Y$ and an *update* map $u : X \times Y' \to X'$. The definition of these pairs $(v, u) : (X, X') \to (Y, Y')$ has been broadened in several directions, including optics and profunctor optics [132, 46] (see also section 2.5.4 later), delta lenses [54, 95], weighted

[4] 'Pointed' in the sense of pointed objects in **Cat**, not in the sense of categories having zero objects

optics [69], and generalized lenses [151], among others. This should not be taken as an exhaustive account of the state of research in area.

We focus on the last, reviewing how the Grothendieck construction can be used to define generalized lenses and charts, and how these structures relate to fibrations. By considering covariant and contravariant functors into **Cat**, as well as their fibrewise opposites, we obtain a flexible language of processes, as introduced in definition 2.3.6. We refer to [92, Ch.1, Ch.9] for a comprehensive introduction to fibred categories and to [79] for a summary on the study of lenses in computer science.

Definition 2.3.1 (Category of elements)**.** Given a functor $F : \mathcal{C} \to \mathbf{Set}$, define its (covariant) category of elements $\mathbf{El}(F)$ having as objects pairs (c, x) with $c \in \mathcal{C}$ and $x \in Fc$. A morphism $(c, x) \to (d, y)$ is a morphism $f : c \to d$ such that $(Ff)(x) = y$ in Fd.

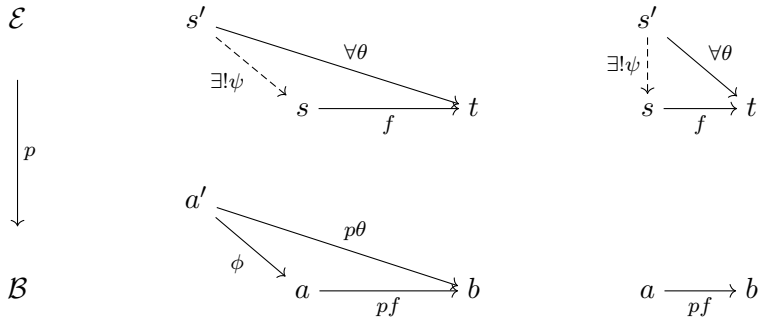
Example 2.3.2. The theory of algebraic databases [139, 142] illustrates categories of elements. A simplified rendition describes database schemas as corresponding to (small) categories $\mathcal{C}$, where objects are datatypes and morphisms represent functional relationships. Database instances correspond to functors $F : \mathcal{C} \to \mathbf{Set}$, where each object of $\mathcal{C}$ maps to the set of data elements, as rows or records in a table. Functional relationships (morphisms in $\mathcal{C}$) are mapped to actual functions between the corresponding sets. The category of elements of an instance F integrates both the schema structure and the data into a single structure.

An analogous definition for contravariant functors has morphisms $(c, x) \to (d, y)$ given by $f : c \to d$ s.t. $(Ff)(y) = x$ in Fc. When the functor indexes not sets but categories, the corresponding construction requires F to be a pseudofunctor to the bicategory **Cat** of categories, functors and natural transformations. When F is contravariant, it is called an *indexed category* [75, 96].

Definition 2.3.3 (Grothendieck construction)**.** Given an indexed category $F : \mathcal{C}^{\mathsf{op}} \to \mathbf{Cat}$, define the (contravariant) Grothendieck construction $\int F$ having as objects pairs (c, x) with $c \in \mathcal{C}$ and $x \in Fc$. A morphism $(c, x) \to (d, y)$ is a pair (f, g) of a morphism $f : c \to d$ in $\mathcal{C}$ and a morphism $g : y \to (Ff)(x)$ in Fc.

Definition 2.3.4 (Cartesian and weakly cartesian morphisms)**.** Let $p : \mathcal{E} \to \mathcal{B}$ be a functor. A morphism $f : s \to t$ in $\mathcal{E}$ is *cartesian* (with respect to p) if for every $\theta : s' \to t$ such that $p\theta = pf \circ \phi$ in $\mathcal{B}$, θ factorizes too as $\psi \circ f$ for a unique lifting ψ of ϕ.

For f to be *weakly* (or *locally*) *cartesian* (with respect to p), it only requires the unique lifting property for $\phi = \mathrm{id}_a : a \to a$.



$\int F$ comes with a functor $p : \int F \to \mathcal{C}$ that is a fibration, meaning that for all objects $y \in \int F$ and morphisms $g : x_0 \to p(y)$, there is a cartesian morphism $f : x \to y$ such that $P(f) = g$. $\int F$ is said to be a *fibred category*. This universal property induces a canonical way to 're-index' or 'pull back' objects and morphisms from the fibre over b to the fibre over a. Moreover, the pseudofunctoriality of the indexing functor F ensures that these pullbacks have coherent structure: pulling back along the identity on a fibre is isomorphic to the identity functor on the fibre, and pulling back along two functors in sequence is isomorphic to pulling back along one after another. These are the associativity and identity laws that are so prevalent in category theory.

Example 2.3.5. Let **Pos** be the 2-category of posets, monotone functions and pointwise inequalities between monotone functions. For a pseudofunctor $F : \mathcal{C}^{\mathsf{op}} \to \mathbf{Pos}$, the functor $p : \int F \to \mathcal{C}$ given by $p(c, x) = c$ is a fibration.

The Grothendieck construction allows a monoidal product that is tied to the pseudofunctor into **Cat** to be lax monoidal [123]. In particular, for $(\mathcal{M}, \otimes, I)$ a symmetric monoidal category, a symmetric lax monoidal pseudofunctor $W : (\mathcal{M}, \otimes, I) \to$

$(\mathbf{Cat}, \times, 1)$ defines a symmetric monoidal product on $\int W$ given by

$$(M, w_M) \otimes (N, w_N) = (M \otimes N, w_M \nabla w_N) \tag{2.1}$$

where $\nabla : W(M) \times W(N) \to W(M \otimes N)$ is the laxator for W.

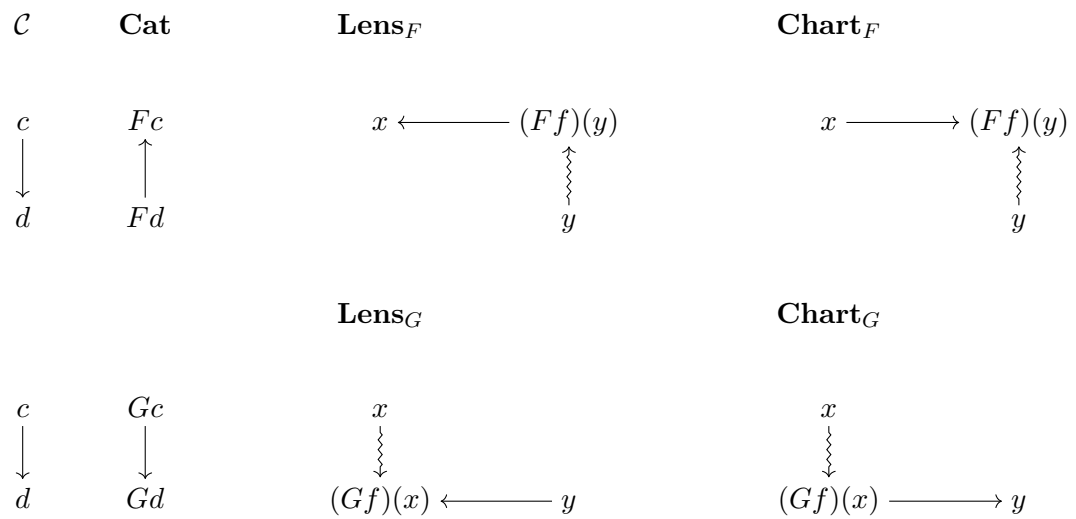
Generalized lenses [151] and 'generalized charts' (whose definition comes from a combination of Spivak's generalized lenses and Jaz Myers' simply-typed charts [125]) are four variants of the construction above, given by op-ing the base category and/or the fibres.

Definition 2.3.6. Given a functors $F : \mathcal{C}^{\mathsf{op}} \to \mathbf{Cat}$ and $G : \mathcal{C} \to \mathbf{Cat}$, the categories $\mathbf{Lens}_F$, $\mathbf{Lens}_G$ of (resp. contravariant and covariant) F-lenses [151], and the categories $\mathbf{Chart}_F$ and $\mathbf{Chart}_G$ of (resp. contravariant and covariant) F-charts have pairs (c, x) with $c \in \mathcal{C}$ and $x \in Fc$ (resp. $x \in Gc$) as objects. Morphisms $(f, g) : (c, x) \to (d, y)$ in all four categories have $f : c \to d$, but their second component g differs:

- In $\mathbf{Lens}_F$, $g : (Ff)(y) \to x$ in Fc.
- In $\mathbf{Lens}_G$, $g : y \to (Gf)(x)$ in Gd.
- In $\mathbf{Chart}_F$, $g : x \to (Ff)(y)$ in Fc.
- In $\mathbf{Chart}_G$, $g : (Gf)(x) \to y$ in Gd.

We illustrate the morphisms of all four variants below. It should be noted that the squiggly arrows are assignments of objects given by functors (e.g. x to $(Ff)(x)$), and

should not be confused with the horizontal arrows, which are morphisms in the fibres.



Consider for any indexed category $F : \mathcal{C}^{\mathsf{op}} \to \mathbf{Cat}$ its fibrewise opposite functor $F^\vee : \mathcal{C}^{\mathsf{op}} \to \mathbf{Cat}$, with $F^\vee c = (Fc)^{\mathsf{op}}$ for $x \in \mathcal{C}$ and $F^\vee f = Ff$ for $f : x \to y$. Note that while the fibres are opped, the functors between them are not, thus $F^\vee$ keeps the same variance as F. This allows to witness the dualities $\mathbf{Chart}_F \cong \mathbf{Lens}_{F^\vee}$ and $\mathbf{Lens}_G \cong \mathbf{Chart}_{G^\vee}$, which are useful in simplifying the notation of bidirectional processes.

Notation 2.3.7. When referring to morphisms in a Grothendieck construction, we will usually draw the diagram with the base component in the bottom and the fibre component in the top. For example, $(c, x) \to (d, y)$ in $\mathbf{Lens}_F$ for $F : \mathcal{C}^{\mathsf{op}} \to \mathbf{Cat}$ will be illustrated as

The categories on the left label where objects and morphisms on the right are. When the upper category is fibered over the lower one, the arrow p shows the fibration. If clear from context, we may omit the labelling of the categories.

2.3.1 Extra structure

Concrete categories, as discussed at the beginning of this section, are understood to be those with a faithful forgetful functor into **Set**. Their objects are 'sets with extra structure' [129], and morphisms that preserve such structure.

Similarly, we may refer to 'categories with extra structure' here informally, as categories that have additional structure [10]. For example, a strict[5] monoidal category is both a category (by its identity and composition satisfying certain laws) and a monoid (by its 'monoidal product' bifunctor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ and monoidal unit I satisfying coherence laws). Likewise, a monoidal category $\mathcal{M}$ has an associated one-object bicategory $\mathbb{B}\mathcal{M}$ obtained via the delooping operation (definition A.1.2).

For these sets (resp. categories) with extra structure, the obvious way to perform the constructions introduced at the beginning of this section is by post-composing by the forgetful functor into **Set** (resp. **Cat**). However, this loses the extra structure that the objects (resp. elements) of the fibres would have, and thus motivates the following definition.

Definition 2.3.8. Let $\mathcal{D}$ be a concrete category, and $F : \mathcal{C} \to \mathcal{D}$ a functor from $\mathcal{C}$ to $\mathcal{D}$. The *category of structured elements* of F, denoted by the same[6] $\mathbf{El}(F)$, consists of objects being pairs $(c : \mathcal{C}, x : Fc)$, whose second component is an element of an object of $\mathcal{D}$, not only of a set. A morphism $(c, x) \to (d, y)$ is a morphism $f : c \to d$ in the indexing category such that $(Ff)(x) = y$ in $\mathcal{D}$, where Ff preserves the structure of x.

This is different from $\int F$ for $F : \mathcal{C} \to \mathbf{Cat}$, as $\mathcal{D}$ is not **Cat** in general, and most notably also from $\mathbf{El}(F)$ for $F : \mathcal{C} \to \mathbf{Set}$, as $\mathcal{D}$ being a concrete category has as objects sets *with structure*, and morphisms that preserve such structure. Consider the following example to see the difference with the category of elements construction $\mathbf{El}(U \circ F)$ applied to F composed with the forgetful functor $U : \mathcal{D} \to \mathbf{Set}$.

Example 2.3.9. Let $\mathcal{I}$ be the category of two objects and a morphism $! : 0 \to 1$ between them, and the concrete category **Grp** of groups and group homomorphisms. Let F :

[5] *Strict* refers to the associator and unitors of the monoidal structure definition A.1.1 being identity natural transformations.

[6] We choose not to introduce new notation, as sets are 'sets with trivial structure'; categories of structured elements for functors into **Set** coincide with categories of elements.

$\mathcal{I} \to \mathbf{Grp}$ be a functor that picks a group homomorphism $h : G \to H$. The second component of a morphism $(!, h) : (0, G) \to (1, H)$ in $\mathbf{El}(F)$ is a group homomorphism, not just a function. In contrast, the second component of $(!, Uh) : (0, UG) \to (1, UH)$ in $\mathbf{El}(U \circ F)$ is a function $Uh : UG \to UH$ between the underlying sets of the groups G and H.

2.4 String diagrams

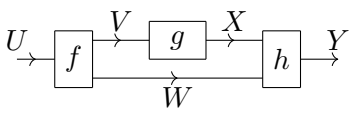
String diagrams [97, 143] are a graphical calculus for representing morphisms in monoidal categories. It is a *formal* graphical representation, where associativity and other laws align with the topological equivalence that their representation on the 2D plane has and makes the manipulation of these diagrams intuitive.

To be more precise, the objects in a monoidal category $(\mathcal{C}, \otimes, I)$ are represented by wires, and morphisms by boxes. Composition is depicted by connecting wires, and the monoidal product by placing diagrams side-by-side. The coherence axioms of monoidal categories (associativity, unit laws) translate into topological equivalences of diagrams.

Example 2.4.1. Let $f : U \to V \oplus W$, $g : V \to X$ and $h : X \oplus W \to Y$ be some linear maps where U, V, W, X, Y are some finite-dimensional vector spaces. If one wanted to express the operation of applying f to vectors in U, then applying g to the V component of the result and finally h to the direct sum of the results in X and W, it would be expressed equationally as

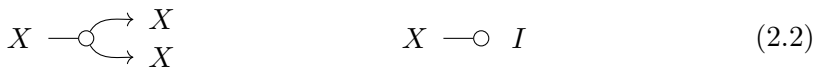
$$U \xrightarrow{f} V \oplus W \xrightarrow{g \oplus W} X \oplus W \xrightarrow{h} Y$$

Because the category of finite dimensional vector spaces is monoidal with direct sum $(\mathbf{FVec}, \oplus, k^1)$, this can also be written in string diagram notation, as



This diagram shows the composition $f \fatsemi (g \oplus W) \fatsemi h : U \to Y$. The direct sum $V \oplus W$ is represented by two wires coming out of the box for f, and similarly for $X \oplus W$.

More specialised string diagrams may introduce graphical symbols for other algebraic operations. For example, symmetric monoidal categories $(\mathcal{C}, \otimes, I)$ with a copy-delete structure [45] have two maps for every object X: 'copy' $\Delta_X : X \to X \otimes X$ and 'delete' $!_X : X \to I$, such that $(!_X, \Delta_X)$ forms a commutative monoid on X compatible with the monoidal structure $(\otimes, I)$. We choose to represent them in string diagrams by white blobs, as show below (some authors prefer black dots).



(2.2)

Examples of copy-delete categories are $(\mathbf{Set}, \times, 1)$, $(\mathbf{FVec}, +, 0)$ (see e.g. later in proposition 4.3.8) and Markov categories [45, 67].

Another class of string diagrams is of bidirectional maps, which we will discuss later in section 2.5.4 once we introduce the definition of *optics*. On that topic, one last string diagram notation that will appear there (fig. 2.4) is that of autonomisations of monoidal categories. A symmetric autonomous category is a symmetric monoidal category such that for all objects A there are 'adjoints' or 'dual objects' A^*, for which cups $\varepsilon : A \otimes A^* \to I$ and caps $\eta : I \to A^* \otimes A$ are freely added and satisfy some equations. Representing objects A with left-to-right wires, their duals A^* are represented with right-to-left-wires, as shown below.

The autonomisation [52] of the cartesian monoidal category $(\mathbf{Set}, \times, 1)$, denoted $L\mathbf{Set}$, has a monoidal product $\otimes$ that does not coincide with the product $\times$ in $\mathbf{Set}$, which prevents $L\mathbf{Set}$ from having all objects be isomorphic to the monoidal unit.

2.5 Categorical cybernetics

In this section we quickly recall the main ideas of categorical cybernetics, mostly from [42], and write down folklore knowledge on biparametrised morphisms and their relation with (co)spans.

2.5.1 Actegories

Given a monoidal category $\mathcal{M}$ and a category $\mathcal{C}$, an action of $\mathcal{M}$ on $\mathcal{C}$, also called an **actegory**, is a functor $\bullet : \mathcal{M} \times \mathcal{C} \to \mathcal{C}$ together with coherence isomorphisms $I \bullet X \cong X$ and $(M \otimes N) \bullet X \cong M \bullet (N \bullet X)$ [41]. Every monoidal category has a self-action given by $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$.

If $\mathcal{M}$ and $\mathcal{C}$ are monoidal categories and $F : \mathcal{M} \to \mathcal{C}$ 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 $\mathcal{C} \to \mathcal{C}$. All actegories in what follows will be symmetric.

An **enrichment** of a category $\mathcal{C}$ in a monoidal category $\mathcal{M}$ is a functor $[-,-] : \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{M}$ plus additional data and conditions. There is a tight connection between actegories and enrichments: if $\bullet$ is any actegory such that every $- \bullet X : \mathcal{M} \to \mathcal{C}$ has a right adjoint $[X, -] : \mathcal{C} \to \mathcal{M}$ (called a closed actegory [93, 41]) 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. An example of such a copowered enrichment is given by $\mathcal{M} = \mathbf{Set}$ next.

Proposition 2.5.1. *Let $\mathcal{C}$ be a category with coproducts. $\mathcal{C}$ has a tensored enrichment in the cartesian monoidal category $(\mathbf{Set}, \times, 1)$ and therefore an action $\bullet : \mathbf{Set} \times \mathcal{C} \to \mathcal{C}$ given by $M \bullet X = \sum_M X$ [41, Ex.3.2.6]. The coherence isomorphisms are given by $1 \bullet X = \sum_1 X \cong X$ and $\sum_{M \times N} X \cong \sum_M \sum_N X$.*

Example 2.5.2. The category $\mathbf{FVec}_k$ has direct sums as coproducts, and is enriched in $\mathbf{Set}$. Writing V for an arbitrary vector space in (the concrete category) $\mathbf{FVec}_k$ and X for an arbitrary set, we have that this enrichment is a *copowered* enrichment in $(\mathbf{Set}, \times, 1)$, as every $[V,-] : \mathbf{FVec}_k \to \mathbf{Set}$ has a left adjoint $- \bullet V : \mathbf{Set} \to \mathbf{FVec}_k$ given by $X \mapsto \underbrace{V \oplus V \oplus \cdots \oplus V}_{|X| \text{ times}}$.

2.5.2 Parametrisation

Given an actegory $\bullet : \mathcal{M} \times \mathcal{C} \to \mathcal{C}$, a **parametrised morphism** $f : X \to Y$ in $\mathcal{C}$ 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 $\alpha : M \to N$ in $\mathcal{M}$ such that $f = g \circ (\alpha \bullet X)$.

Definition 2.5.3. Let $\bullet : \mathcal{M} \times \mathcal{C} \to \mathcal{C}$ be an actegory. We denote by $\mathbf{Para}_{\mathcal{M}}(\mathcal{C})$ or $\mathbf{Para}_{\bullet}(\mathcal{C})$ the bicategory whose objects are objects of $\mathcal{C}$, 1-cells are parametrised morphisms and 2-cells are reparametrisations.

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 bicategory $\mathbf{Copara}_{\mathcal{M}}(\mathcal{C})$ of objects, coparametrised morphisms and reparametrisations.

A morphism $f : X \to Y$ that is both parametrised and coparametrised is called a **biparametrised morphism**, and consists of two objects $M, N : \mathcal{M}$ and a morphism $f : M \bullet X \to N \bullet Y$. A reparametrisation from (M, N, f) to (M', N', g) is a pair of morphisms $\alpha : M \to M'$ and $\beta : N \to N'$ in $\mathcal{M}$ such that $f = (\beta \bullet Y) \circ g \circ (\alpha \bullet X)$.

Definition 2.5.4. Let $\bullet : \mathcal{M} \times \mathcal{C} \to \mathcal{C}$ be an actegory. We denote by $\mathbf{Bipara}_{\mathcal{M}}(\mathcal{C})$ or $\mathbf{Bipara}_{\bullet}(\mathcal{C})$ the bicategory whose objects are objects of $\mathcal{C}$, 1-cells are biparametrised morphisms and 2-cells are reparametrisations.

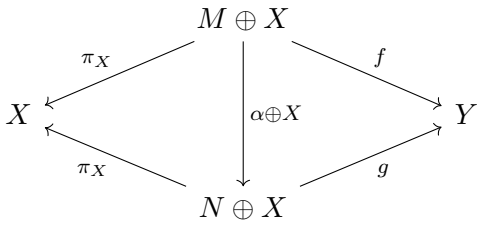
Definition 2.5.5. Let $\mathcal{C}$ be a category enriched in a monoidal category $\mathcal{M}$. An **externally parametrised morphism** $f : X \to Y$ of $\mathcal{C}$ is a pair of an object M of $\mathcal{M}$ and a morphism $f : M \to [X, Y]$ of $\mathcal{M}$ [148]. There is once again a bicategory $\mathbf{Para}_{\mathcal{M}}(\mathcal{C})$ of externally parametrised morphisms.

In the case of a tensored enrichment this bicategory is equivalent to definition 2.5.3, but there are also interesting cases when they differ. Coparametrised morphisms cannot be defined for an enrichment that is not tensored.

In certain cases, (co)parametrised morphisms coincide with (co)spans. This has been explored in great generality in [39]. For our purposes, we focus on their relation for categories with biproducts.

Proposition 2.5.6. *Let $\mathcal{C}$ be a category whose monoidal product and self-action are given by a biproduct $\oplus$ (i.e. $M \bullet X := M \oplus X$). Let $\mathbf{Span}(\mathcal{C})$ be the bicategory of objects of $\mathcal{C}$, spans $X \leftarrow N \rightarrow Y$ as 1-cells and 2-cells given by $M \rightarrow N$ [18] (see also [50]). Then there are wide embeddings of bicategories $\mathbf{Para}_{\oplus}(\mathcal{C}) \hookrightarrow \mathbf{Span}(\mathcal{C})$ and $\mathbf{Copara}_{\oplus}(\mathcal{C}) \hookrightarrow \mathbf{Cospan}(\mathcal{C})$.*

Proof. All objects of $\mathbf{Span}(\mathcal{C})$ are present in $\mathbf{Para}(\mathcal{C})$. A morphism $X \rightarrow Y$ in $\mathbf{Para}_{\oplus}(\mathcal{C})$ consists of $f : M \oplus X \rightarrow Y$ in $\mathcal{C}$. This defines a morphism $X \overset{\pi_X}{\leftarrow} M \oplus X \overset{f}{\rightarrow} Y$ in $\mathbf{Span}(\mathcal{C})$. Composition of morphisms in $\mathbf{Para}(\mathcal{C})$ coincides with composition of morphisms $X \leftarrow M \oplus X \rightarrow Y$ in $\mathbf{Span}(\mathcal{C})$, as the pullback with one leg being a projection results in products. Given $f : M \oplus X \rightarrow Y$ and $g : N \oplus X \rightarrow Y$, a reparametrisation $\alpha : M \rightarrow N$ corresponds to a morphism $\alpha \oplus X$ between spans



The mapping of coparametrised morphisms $f : X \rightarrow N \oplus Y$ and their composition to cospans $X \overset{f}{\rightarrow} N \oplus Y \overset{\iota_Y}{\leftarrow} Y$ and pushouts is similar. □

Morphisms $(M, N, f) : X \rightarrow Y$ in $\mathbf{Bipara}_{\oplus}(\mathcal{C})$ decompose into 'block matrices', i.e. tuples (M, N, a, b, c, d) with $a : X \rightarrow Y$, $b : M \rightarrow Y$, $c : X \rightarrow N$, $d : M \rightarrow N$ in $\mathcal{C}$, denoted

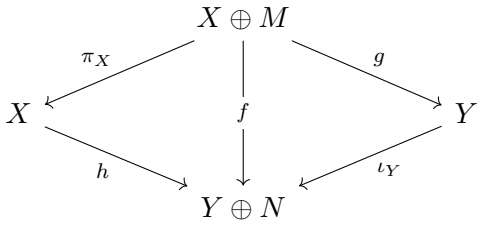
$$f = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

This 'block-decomposition' of biparametrised morphisms is given by the matrix representation lemma for categories with biproducts [90, Lem.2.26], and gives explicit expressions to the maps g and h next.

Proposition 2.5.7. $\mathbf{Bipara}_{\oplus}(\mathcal{C})$ *is isomorphic to a subcategory of* $\mathbf{Span}(\mathcal{C}) \times \mathbf{Cospan}(\mathcal{C})$.

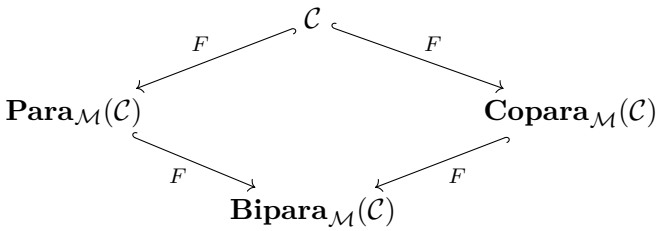
Proof. $\mathbf{Span}(\mathcal{C}) \times \mathbf{Cospan}(\mathcal{C})$ is a product category; its objects are pairs $X \times Y$ where $X : \mathbf{Span}(\mathcal{C})$ and $Y : \mathbf{Cospan}(\mathcal{C})$, i.e. pairs of objects of the underlying category $\mathcal{C}$. Morphisms are pairs of a span and a cospan in $\mathcal{C}$.

On objects, X maps to $X \times X$. A morphism $(M, N, f) : X \to Y$ in $\mathbf{Bipara}_{\oplus}(\mathcal{C})$ corresponds to a pair $(X \overset{\pi_X}{\leftarrow} X \times M \overset{g}{\to} Y) \times (X \overset{h}{\to} N \times Y \overset{\iota_Y}{\leftarrow} Y)$ such that the following commutes.



□

Proposition 2.5.8. *There are the following embeddings given by strict functors, all labelled as* F *when clear from context.*



Proof. $\mathbf{Para}_{\mathcal{M}}(\mathcal{C}) \hookrightarrow \mathbf{Bipara}_{\mathcal{M}}(\mathcal{C})$ is given by identity on objects, $M \bullet X \to Y$ to $M \bullet X \to I \bullet Y$ on morphisms and $h : M \to M'$ to (h, id_I) on 2-cells, and similarly for $\mathbf{Copara}_M(\mathcal{C}) \hookrightarrow \mathbf{Bipara}_{\mathcal{M}}(\mathcal{C})$. See fig. 2.1 for an illustration of the embeddings for morphisms. □

The free functor $F : \mathbf{Set} \to \mathbf{Cat}$ that takes the indiscrete category construction over sets has a left adjoint $U : \mathbf{Cat} \to \mathbf{Set}$ (forgetful functor) which maps categories to isomorphism classes of objects, and a right adjoint $\pi_0 : \mathbf{Cat} \to \mathbf{Set}$ (connected components functor). Change of enrichment along U (resp. π_0) turns any bicategory into a 1-category by quotienting together 1-cells that are related by invertible (resp. any) 2-cell.

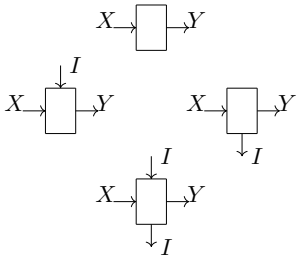


Figure 2.1: Embedding of a morphism in $\mathcal{C}$ (top) into $\mathbf{Para}_{\mathcal{M}}(\mathcal{C})$ (left), $\mathbf{Copara}_{\mathcal{M}}(\mathcal{C})$ (right), and $\mathbf{Bipara}_{\mathcal{M}}(\mathcal{C})$ (bottom).

Example 2.5.9. The morphisms in the 1-category $\pi_0^*(\mathbf{Para}_{\mathcal{M}}(\mathcal{C}))$ are equivalence classes identifying any two parameters $M, N \in \mathcal{M}$ such that there exists $\alpha : M \to N$ in $\mathcal{M}$. On the other hand, the 1-category $U^*(\mathbf{Para}_{\mathcal{M}}(\mathcal{C}))$ does distinguish between morphisms $(M, f : M \bullet X \to Y)$ and $(N, g : N \bullet X \to Y)$ for which M and N are not isomorphic objects of $\mathcal{M}$. Similar 1-categories may be obtained from $\mathbf{Copara}_{\mathcal{M}}(\mathcal{C})$ and $\mathbf{Bipara}_{\mathcal{M}}(\mathcal{C})$.

Proposition 2.5.10. *Let $\mathcal{M}$ be a monoidal category with monoidal unit I.*

i. If I is initial, $F : \mathcal{C} \hookrightarrow \mathbf{Para}_{\mathcal{M}}(\mathcal{C})$ and $F : \mathbf{Copara}_{\mathcal{M}}(\mathcal{C}) \hookrightarrow \mathbf{Bipara}_{\mathcal{M}}(\mathcal{C})$ have left inverse functors.

ii. If I is terminal, $F : \mathcal{C} \hookrightarrow \mathbf{Copara}_{\mathcal{M}}(\mathcal{C})$ and $F : \mathbf{Para}_{\mathcal{M}}(\mathcal{C}) \hookrightarrow \mathbf{Bipara}_{\mathcal{M}}(\mathcal{C})$ have left inverse functors.

iii. If I is a zero object, $F : \mathcal{C} \hookrightarrow \mathbf{Bipara}_{\mathcal{M}}(\mathcal{C})$ has a left inverse functor.

Proof. Let G denote all putative left inverse functors. They are all defined as identity-on-objects. Their definition on morphisms is given next. Let $f : M \bullet X \to Y$ be a morphism in $\mathbf{Para}_{\mathcal{M}}(\mathcal{C})$. Then, $G(f) : X \to Y$ in $\mathcal{C}$ is given by

$$X \xrightarrow{\sim} I \bullet X \xrightarrow{!\bullet X} M \bullet X \xrightarrow{f} Y$$

G is left inverse to $F : \mathcal{C} \to \mathbf{Para}_{\mathcal{M}}(\mathcal{C})$, as $G \circ F : \mathcal{C} \to \mathcal{C}$ is naturally isomorphic to $\mathrm{id}_{\mathcal{C}}$. I being initial is a condition for the well-definition of G; without it, an image of a morphism $M \bullet X \to Y$ in $\mathcal{C}$ cannot be defined.

A zero object I is both initial and final. For $g : M \bullet X \to N \bullet Y$ in $\mathbf{Bipara}_{\mathcal{M}}(\mathcal{C})$,

the map $G(g) : X \to Y$ in $\mathcal{C}$ is given by

$$X \xrightarrow{\sim} I \bullet X \xrightarrow{!\bullet X} M \bullet X \xrightarrow{g} N \bullet Y \xrightarrow{!\bullet Y} I \bullet Y \xrightarrow{\sim} Y$$

Similarly, G is a left inverse to $F : \mathcal{C} \hookrightarrow \mathbf{Bipara}_M(\mathcal{C})$. □

This means that, for common choices of $\mathcal{M}$, working with (co- or bi-)parametrised categories is equivalent by construction to working with $\mathcal{C}$ directly.

2.5.3 Categories of processes

Under some (implicit) semantics of a category $\mathcal{C}$ as having spaces as objects and **processes** as morphisms, $\mathbf{Para}_{\mathcal{M}}(\mathcal{C})$ will capture **controlled processes**, where we associate the parameter M of a morphism $M \bullet X \to Y$ with the control. Similarly, $\mathbf{Copara}_{\mathcal{M}}(\mathcal{C})$ will describe **observed processes**, where a morphism $X \to N \bullet Y$ has as coparameter N the observation. The category $\mathbf{Bipara}_{\mathcal{M}}(\mathcal{C})$ will unify both as **controlled and observed processes**.

In particular, *linear* controlled processes [150, Ch.2][99] are morphisms in $\mathbf{Para}_{+}(\mathbf{FVec})$ and *linear* observed processes [150, Ch.2][99] are morphisms in $\mathbf{Copara}_{+}(\mathbf{FVec})$.

Example 2.5.11. *Linear* controlled and observed processes are morphisms in $\mathbf{Bipara}_{+}(\mathbf{FVec}_k)$ for $k \in \{\mathbb{C}, \mathbb{R}\}$ and $\mathbf{Bipara}_{+}(\mathbf{FMod}_R)$ for a ring R, where the self action $+$ for both is the biproduct. Even though there is not a single abstract definition of (discrete) **state space models** (SSM), their common use in control theory [99, 169] is captured by endomorphisms in $\mathbf{Bipara}_{+}(\mathbf{FVec})$.

- In control theory, the linear dynamics of a controlled system with state space X, control space U and observation space Y are specified by two equations that define a morphism $(U, Y, A, B, C, D) : X \to X$:

$$\boldsymbol{x}' = A\boldsymbol{x} + B\boldsymbol{u}$$
$$\boldsymbol{y} = C\boldsymbol{x} + D\boldsymbol{u}$$

where we write $\boldsymbol{x}$ and $\boldsymbol{u}$ for the current state, control and observation vectors, and $\boldsymbol{x}'$ for the next state.

- In machine learning, recurrent neural networks (RNN) [91] process sequential data by retaining internal memory in the form of a hidden state H in addition to the input space X, output space Y, weight matrix W and bias vector b. The hidden state update equation is typically given by $h_t = \sigma(U \cdot X + W \cdot h_{t-1} + b)$, for some nonlinear function σ. They have been characterised via *delayed traces* in [153] by taking the input and output spaces as first class objects of a (strict cartesian) category $\mathcal{C}$ and the hidden state in the causal extension of the category $\mathrm{St}(\mathcal{C})$: given an initial hidden state h_0, a morphism from an input sequence $\{X_i\}$ to an output sequence $\{Y_i\}$ is given by the (extensional equivalence class of) the unrolling of the hidden state update and output calculation across the input sequence. By choosing H to be first class instead, we can consider the RNN equations as biparametrised endomorphisms $(X, Y, (W, b)) : H \to H$. Our understanding of RNNs is not at all dissimilar to their approach, and providing a closer relation to their *state construction* [153] and signal flow graphs [31] is future work.

Other examples of linear processes for $\mathcal{C} = \mathbf{FVec}$ and the justification for the need of definition 2.5.4 will be shown in section 4.3.

2.5.4 Optics

Having discussed F-lenses and their application for bidirectional processes in section 2.3, we focus now on another generalization of lenses called *optics*. These appear for example in applications to probability: consider the hypothetical definition of a lens between pairs of measurable spaces $(X, X') \to (Y, Y')$ whose forward and backward maps are Markov kernels. The image of the forward map in its codomain is a distribution over $X \times Y$ that is not disjoint, thus the composition with $(Y, Y') \to (Z, Z')$ is not the composition of lenses. Example 2.5.16 will revisit how these are modelled with optics. We will start with the particular definition of *mixed optics*, and see how regular optics and lenses appear as particular cases of it.

Definition 2.5.12 (Mixed optic, cf. [46])**.** Let $\mathcal{M}$ be a monoidal category acting on categories $\mathcal{C}$ and $\mathcal{D}$. The category $\mathbf{Optic}_{\mathcal{M}}(\mathcal{C}, \mathcal{D})$ has pairs $(X : \mathcal{C}, Y : \mathcal{D})$ as objects. Given objects X, Y of $\mathcal{C}$ 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 $M : \mathcal{M}$ called the *residual*, a coparametrised morphism $f : X \to M \bullet Y$ of $\mathcal{C}$ and a parametrised morphism $f' : M \bullet Y' \to X'$ of $\mathcal{D}$. Hom-sets consist of equivalence classes that are generated by reparametrisations and satisfy the universal property of a coend in **Set**:

$$\mathbf{Optic}_{\mathcal{M}}(\mathcal{C}, \mathcal{D}) \left(\binom{X}{X'}, \binom{Y}{Y'} \right) = \int^{M:\mathcal{M}} \mathcal{C}(X, M \bullet Y) \times \mathcal{D}(M \bullet Y', X') \tag{2.3}$$

When both actions are symmetric, or equivalently are defined by a span of symmetric monoidal functors $\mathcal{C} \leftarrow \mathcal{M} \to \mathcal{D}$, then $\mathbf{Optic}_{\mathcal{M}}(\mathcal{C}, \mathcal{D})$ is a symmetric monoidal category, with the tensor product on objects being pairwise [41].

In the common case that $\mathcal{M} = \mathcal{C} = \mathcal{D}$ acts on itself by monoidal product, we get the original definition of optics.

Definition 2.5.13 ([136])**.** Let $\mathcal{M} = \mathcal{C} = \mathcal{D}$, and both actions $\bullet : \mathcal{M} \times \mathcal{C} \to \mathcal{C}$, $\bullet : \mathcal{M} \times \mathcal{D} \to \mathcal{D}$ be given by tensor product $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$. The category $\mathbf{Optic}(\mathcal{C})$ has pairs $(X : \mathcal{C}, Y : \mathcal{C})$ as objects, and morphisms are

$$\mathbf{Optic}(\mathcal{C}) \left(\binom{X}{X'}, \binom{Y}{Y} \right) = \int^{M:\mathcal{C}} \mathcal{C}(X, M \otimes Y) \times \mathcal{C}(M \otimes Y', X') \tag{2.4}$$

The tensor product of $\mathbf{Optic}(\mathcal{C})$ is pairwise monoidal product.

Continuing the topic of section 2.4, optics forming a (symmetric) monoidal category allow the usual string diagrammatic syntax where wires are objects (X, X') and a box from (X, X') to (Y, Y') denotes an optic (M, f, f') with that domain and codomain (fig. 2.2, left). In addition, there are two specialised string diagram notations for $\mathbf{Optic}(\mathcal{C})$ that show the underlying morphisms in $\mathcal{C}$ too. The first (fig. 2.2, middle) considers them as morphisms of a monoidal category [82], composing left-to-right, with causality flowing clockwise from top-left. 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 optic can be read from a diagram, as the monoidal product of the wire labels of all right-to-left bending wires. These diagrams have only been properly formalised for a monoidal category acting on itself, so for mixed optics we are technically being informal. The second (fig. 2.2, right) considers them as colours of an operad [87][168, Ch.2], composing outside-in, with causality flowing left-to-right.

For example, fig. 2.2(top) shows a typical optic $(M, f, f') \in \mathbf{Optic}(\mathcal{C})\left(\binom{X}{X'}, \binom{Y}{Y'}\right)$ represented in the three notations.

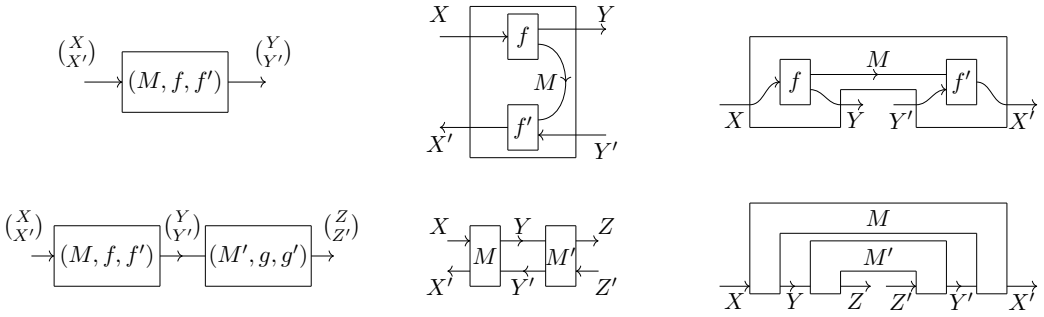


Figure 2.2: Usual monoidal (left), 'bidirectional' (middle) and 'comb' [138, §4.3][86] (right) string diagram notations for morphisms in $\mathbf{Optic}(\mathcal{C})$ (top) and their composition (bottom).

There are two common cases when the coend in the definition of mixed optics (2.3) can be eliminated using the ninja Yoneda lemma [136, 113]. Firstly, when $\mathcal{M} = \mathcal{C} = \mathcal{D}$ acts on itself by *cartesian* product then there is a natural isomorphism

$$\int^{M:\mathcal{C}} \mathcal{C}(X, M \times Y) \times \mathcal{C}(M \times Y', X') \cong \int^{M:\mathcal{C}} \mathcal{C}(X, M) \times \mathcal{C}(X, Y) \times \mathcal{C}(M \times Y', X')$$
$$\cong \mathcal{C}(X, Y) \times \mathcal{C}(X \times Y', X')$$

This is usually known as a (concrete) **Lens**, as introduced earlier in section 2.3. Their bidirectional diagrammatic syntax is shown in fig. 2.3. Although this case is much easier to understand, there are significant conceptual advantages to the more general definition [70]. Secondly, when $\mathcal{M} = \mathcal{C} = \mathcal{D}$ acts on itself by a closed monoidal product then there is a natural isomorphism $\mathbf{Optic}(\mathcal{C})\left(\binom{X}{X'}, \binom{Y}{Y'}\right) \cong \mathcal{C}(X, Y \otimes [Y', X'])$ which produces *linear lenses* [136, p. 4.8]. Both of these cases can be generalised to requiring a condition on only one side.

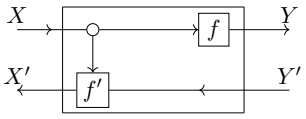


Figure 2.3: 'Bidirectional' string diagram notation for a morphism in **Lens**($\mathcal{C}$). Note that the duplication of the input is explicit by the copy map Δ_X drawn as the white blob (2.2). The drawing of the backward map as a morphism with an input from the top is merely a stylistic choice, where it should be understood as a morphism with two inputs.

Example 2.5.14. Let **Set** act on itself by cartesian product. Optics $\binom{X}{X'} \to \binom{Y}{Y'}$ in **Optic**(**Set**) can be written equivalently as pairs of functions $X \to Y$ and $X \times Y' \to X'$, or as a single function $X \to Y \times (Y' \to X')$. For the cartesian self-action of **Set**, **Optic**(**Set**) coincides with the category of monomial endofunctors (those of the form $F(X) = A \times X^B$) and natural transformations.

Example 2.5.15. Optics $\binom{X}{X'} \to \binom{Y}{Y'}$ in **Optic**($\mathbf{Mfd}_\infty$) can be written as pairs of smooth functions $X \to Y$ and $X \times Y' \to X'$ between manifolds. When the manifolds are globally diffeomorphic to $\mathbb{R}^n$ for some n, these optics are defined in **Optic**(**Smooth**).

Example 2.5.16 ([85], cf. effectful optics [136])**.** 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 [67], and it is neither cartesian monoidal nor monoidal closed. Optics $\binom{X}{X'} \to \binom{Y}{Y'}$ in **Optic**(**Mark**) can only be written as optics, it is not possible to eliminate the coend. This is the setting used for Bayesian open games [27].

Example 2.5.17 ([85])**.** Let **Conv** be the category of convex sets, which is the Eilenberg-Moore category of the finite support probability monad [68]. A convex set can be thought of a set with an abstract expectation operator $\mathbb{E} : \Delta X \to X$. Thus the functor $\Delta : \mathbf{Mark} \hookrightarrow \mathbf{Conv}$ of free algebras as algebras 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 [156, section 2.2] (see also [106]).

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 $M \bullet X = \Delta(M) \otimes X$ [41, section 5.5]. Together with the self-action of **Mark**, we get a category $\mathbf{Optic}(\mathbf{Mark}, \mathbf{Conv})$ given by

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

(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 in section 3.4 for Markov decision processes, where the forwards direction is a Markov kernel and the backwards direction is a function involving expectations.

Any cartesian self-action in a category with finite limits can be generalised to **dependent lenses** (also known as morphisms of **containers** [1]), which in the locally cartesian closed case are equivalent to **polynomial endofunctors** [128]. Finding the best of both worlds between the monoidal and cartesian cases is known as **dependent optics** [161] and is only partially understood. It is common that the available actions of a reinforcement learning agent depends on the current state of the Markov chain [34] (see also example 4.3.3), and dependent optics are believed to be a way to formalise this dependency. In chapter 3 we only consider the simply-typed case, and we postpone an alternative formalisation in terms of spans (current state space ← current action space indexed by state → next state space) to chapter 4.

Definition 2.5.18 (States and continuations)**.** Let $\mathcal{C}$ be a category with a monoidal unit I that is terminal. By abuse of notation, we also write $I = \begin{pmatrix} I \\ I \end{pmatrix}$ for the monoidal unit

of $\mathbf{Optic}(\mathcal{C})$. Then we have two natural isomorphisms:

i. Between *states* in $\mathcal{C}$ and *states* in $\mathbf{Optic}(\mathcal{C})$:

$$\mathbf{Optic}(\mathcal{C})\left(I, \binom{X}{X'}\right) \cong \mathcal{C}(I, X)$$

ii. Between *costates* in $\mathbf{Optic}(\mathcal{C})$ and *continuations* in $\mathcal{C}$:

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

These are obtained as the image of the representable [80] functor, defined as

$$\mathbb{K} = \mathbf{Optic}(\mathcal{C})(-, I) : \mathbf{Optic}(\mathcal{C})^{\mathsf{op}} \to \mathbf{Set} \tag{2.5}$$

Remark 2.5.19. The image of optics in **Set** under $\mathbb{K}$ are formal string diagrams (fig. 2.4) in the free autonomisation [52] of $(\mathbf{Set}, \times, 1)$, denoted $(L\mathbf{Set}, \otimes, I)$ (see section 2.4). The diagrammatic notation for $L\mathbf{Set}$ uses the isomorphism $\lambda : A^* \otimes B \cong B^A$ and the cap $\eta : I \to A^* \otimes A$. Morphisms in **Set** like $\mathrm{ev} : B^A \times A \to B$ embed via the strong monoidal fully faithful functor $F : \mathbf{Set} \to L\mathbf{Set}$.

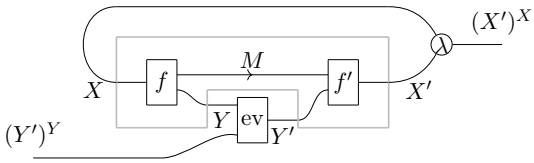


Figure 2.4: Image of a morphism $(M, f, f') : \binom{X}{X'} \to \binom{Y}{Y'}$ in $\mathbf{Optic}(\mathbf{Set})$ under $\mathbb{K}$ in the free autonomisation of **Set**.

Example 2.5.20. Let $\mathcal{C}$ be a Markov category. States into $\binom{X}{X'} : \mathbf{Optic}(\mathcal{C})$ are random variables valued in X, while continuations $\mathbb{K}\binom{X}{X'}$ are Markov kernels $X \to X'$.

Example 2.5.21. Let $\mathcal{C}$ be a cartesian monoidal category. Given a morphism $\binom{f}{f'} : \binom{X}{X'} \to \binom{Y}{Y'}$ in $\mathbf{Optic}(\mathcal{C}) \cong \mathbf{Lens}(\mathcal{C})$ and a morphism $k : Y \to Y'$ in $\mathcal{C}$, the action of $\mathbb{K}$ on $\binom{f}{f'}$ is a morphism $X \to X'$ in $\mathcal{C}$ given by $x \mapsto f'(x, k(f(x)))$.

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

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

When a category of optics is symmetric monoidal, it admits a self-action. In [42] it was shown that this action results in the category $\mathbf{Para}(\mathbf{Optic})$ of **parametrised optics**, whose study was named **categorical cybernetics** and includes applications in compositional game theory [49], deep learning [48], compositional Bayesian inference [37] and variational learning [148], and applications in software engineering such as *open servers* [163].

2.6 Weighted para, decorated cospans and decorated spans

Recall from section 2.3 that for a symmetric lax monoidal functor $(W, \nabla) : (\mathcal{M}, \otimes, I) \to (\mathbf{Set}, \times, 1)$ (resp. $(\mathbf{Cat}, \times, 1)$), the category of elements $\mathbf{El}(W)$ (resp. the Grothendieck construction $\int W$) is symmetric monoidal (2.1).

This symmetric monoidal product allows to extend an action $\bullet : \mathcal{M} \times \mathcal{C} \to \mathcal{C}$ of $\mathcal{M}$ on a symmetric monoidal category $\mathcal{C}$ to an action of $\mathbf{El}(W)$ (resp. $\int W$) on $\mathcal{C}$, by precomposing with the discrete fibration $\pi : \mathbf{El}(W) \to \mathcal{M}$ to obtain $(M, w) \bullet X = M \bullet X$. Therefore a notion of parametrisation by $\mathbf{El}(W)$ arises, where the second component of objects in $\mathbf{El}(W)$ are referred to as 'weights'.

Definition 2.6.1 (Weighted para, cf. weighted copara [69])**.** We write $\mathbf{Para}^W_{\mathcal{M}}(\mathcal{C})$ for $\mathbf{Para}_{\mathbf{El}(W)}(\mathcal{C})$ (resp. $\mathbf{Para}_{\int W}(\mathcal{C})$), and call this *weighted parametrisation*.

Example 2.6.2. Let $\mathcal{C}$ be a symmetric monoidal category and $W : \mathcal{C} \to \mathbf{Set}$ a symmetric lax monoidal functor. Consider the bicategory $\mathbf{Para}^W_{\otimes}(\mathcal{C})$ generated by the action of $\int W$ on $\mathcal{C}$ given by $(M, w) \bullet X = M \otimes X$. A parametrised morphism $X \to Y$ of $\mathcal{C}$ weighted by W consists of a morphism $M \otimes X \to Y$ together with an element $w \in W(M)$, as depicted in fig. 2.5 (left).

The construction of a 1-category out of $\mathbf{Para}^W_{\otimes}(\mathcal{C})$ follows example 2.5.9.

Example 2.6.3. The 1-category $\pi_0^*(\mathbf{Para}^W(\mathcal{C}))$ has morphisms that are equivalence classes identifying all ways of making the cut in fig. 2.5(right), while the 1-category $U^*(\mathbf{Para}^W(\mathcal{C}))$ does distinguish between morphisms $(M, f : M \bullet X \to Y)$ and $(N, g : N \bullet X \to Y)$ for which M and N are not isomorphic objects of $\mathcal{M}$.

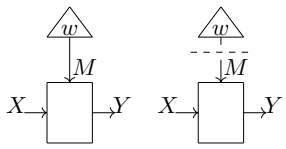


Figure 2.5: Morphism in $\mathbf{Para}_{\otimes}^W(\mathcal{C})$ (left) and its equivalence class in $\pi_0^*(\mathbf{Para}_{\otimes}^W(\mathcal{C}))$ (right).

The 1-category $\pi_0^*(\mathbf{Para}^W(\mathcal{C}))$ satisfies an important universal property: it is the symmetric monoidal category that results from freely extending $\mathcal{C}$ with a state $w : I \to X$ for each element $w \in W(X)$, for all objects X [88][7].

Weighted para categories arise by first defining the weights w over objects M in the acting category $\mathcal{M}$, and then defining the action on $\mathcal{C}$, by forgetting the weight and applying $\bullet$. The weight is thus a 'label' which is purely syntactic and does not change the action, unlike e.g. the action of groups in a vector space.

An alternative construction, which turns out to be useful in chapter 4 to talk about certain algorithms in control theory, is to choose to first apply an action and later define a 'weight' over the resulting object $M \bullet X$. To motivate its definition, we first review the concept of decorated cospans.

Definition 2.6.4 (Decorated cospans, [59])**.** Let $(F, \nabla) : (\mathcal{C}, +, 0) \to (\mathcal{D}, \otimes, 1)$ be a lax monoidal functor with laxator $\nabla_{N,M} : FN \otimes FM \to F(N + M)$ in $\mathcal{D}$. The category $F\mathbf{Cospan}$ of F-cospans consists of objects of $\mathcal{C}$ and morphisms from X to Y being equivalence classes of cospans $X \xrightarrow{i} N \xleftarrow{o} Y$ and 'structure elements' $1 \xrightarrow{s} FN$. Composition of cospans is given by pushouts, and the structure elements $1 \xrightarrow{s} FN$ and $1 \xrightarrow{t} FM$ compose via

$$1 \xrightarrow{\lambda^{-1}} 1 \otimes 1 \xrightarrow{s \otimes t} FN \otimes FM \xrightarrow{\nabla_{N,M}} F(N + M) \xrightarrow{F[j_N, j_M]} F(N +_Y M)$$

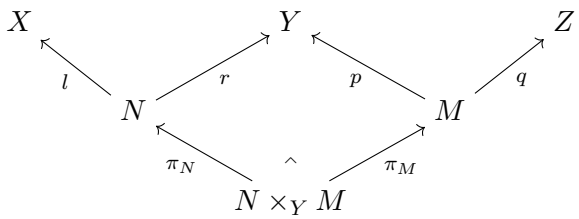
[7] Thanks to Nathan Corbyn for bringing this reference to our attention.

where $j_N : N \to N +_Y M$ and $j_M : M \to N +_Y M$ are the two injections into the pushout.

This definition serves as a common framework to model many classes of open systems: open electrical circuits [11], open Markov processes [12], open Petri nets [13] among others. The usual application considers the apex of the cospan to be indexing some kind of autonomous system, and the legs of the cospan identify some open interfaces under which they compose.

Our approach employs the theory of decorated cospans with a slightly different mental model, by encoding (discrete) dynamics in the legs of spans. This allows the composition by pullback to discard by construction the unreachable states of a trajectory (cf. 'zig-zag diagrams' [14] and reachability in sequential decision problems [34]). We first consider a 'dual' construction of decorated spans, which requires the morphisms in the base category to be spans instead of cospans, and a contravariance in the functor F.

Definition 2.6.5 (Decorated spans)**.** Let $(\mathcal{C}, \times, 1)$ and $(\mathcal{D}, \otimes, I)$ be two monoidal categories, and let $\mathcal{C}$ have pullbacks. Let $(F, \nabla) : (\mathcal{C}, \times, 1)^{\mathsf{op}} \to (\mathcal{D}, \otimes, I)$ be a lax monoidal contravariant functor with laxator $\nabla_{N,M} : FN \otimes FM \to F(N \times M)$ in $\mathcal{D}$. The category $F\mathbf{Span}$ of F-spans consists of objects of $\mathcal{C}$ and morphisms from X to Y being equivalence classes of spans $X \xleftarrow{l} N \xrightarrow{r} Y$ and 'structure elements' $I \xrightarrow{s} FN$ from the comonoid unit to the image of the apex under F. Composition of spans is given by pullbacks in $\mathcal{C}$



and the structure elements $I \xrightarrow{s} FN$ and $I \xrightarrow{t} FM$ compose in $\mathcal{D}$ via

$$I \xrightarrow{\lambda^{-1}} I \otimes I \xrightarrow{s \otimes t} FN \otimes FM \xrightarrow{\nabla_{N,M}} F(N \times M) \xrightarrow{F\langle \pi_N, \pi_M \rangle} F(N \times_Y M) \tag{2.6}$$

where $\pi_N : N \times_Y M \to N$ and $\pi_M : N \times_Y M \to M$ are the two projections from

the pullback, and $\langle \pi_N, \pi_M \rangle : N \times_Y M \to N \times M$ maps under the contravariant F to $F(N \times M) \to F(N \times_Y M)$.

Notation 2.6.6. Following notation 2.3.7, diagrams for decorated spans and decorated cospans will be drawn with the morphisms in $\mathcal{C}$ in the bottom layer and the decoration above the apex. So we illustrate $(i, o, s) : X \to Y$ in $F\mathbf{Cospan}$ and $(l, r, s) : X \to Y$ in $F\mathbf{Span}$ respectively as

$F\mathbf{Cospan}$	s	$F\mathbf{Span}$	s
C	$X \xrightarrow[i]{} N \xleftarrow[o]{} Y$	$\mathcal{C}$	$X \xleftarrow[l]{} N \xrightarrow[r]{} Y$

Parametric spans [20] are a similar construction in the context of neural networks, given by three-legged spans whose apex is fibred over the *middle* leg. Next we demonstrate the decorated span construction for the contravariant powerset functor.

Example 2.6.7. Let $(\mathcal{C}, \times, 1) = (\mathbf{FinSet}, \times, I)$, with pullbacks being the usual subsets of products. Let $(\mathcal{D}, \otimes, I) = (\mathbf{Pos}, \times, I)$ be the category of posets with cartesian monoidal product. The monoidal unit in $\mathcal{D}$ is the singleton poset I with $* \leq *$ and the canonical comonoid map $\lambda^{-1}(*) = (*, *)$. The contravariant powerset functor $\mathcal{P}^{\mathsf{op}} : \mathbf{FinSet}^{\mathsf{op}} \to \mathbf{Pos}$ maps $f : X \to Y$ to $\mathcal{P}^{\mathsf{op}} : \mathcal{P}Y \to \mathcal{P}X$, where a predicate $p : Y \to 2$ is sent to $p \circ f : X \to 2$. $\mathcal{P}^{\mathsf{op}}$ is lax monoidal: the laxator $\nabla_{N,M} : \mathcal{P}(N) \times \mathcal{P}(M) \to \mathcal{P}(N \times M)$ maps a pair of posets $(A \subseteq N, B \subseteq M)$ to their product $A \times B \subseteq N \times M$ with pointwise order, which we write in infix form as $(N \times M, A \nabla B)$. It is not strong as not all sets in $\mathcal{P}(N \times M)$ are a product of posets in $\mathcal{P}(N)$ and $\mathcal{P}(M)$ (consider e.g. $N = \{a, b\}$, $M = \{c, d\}$ and the subset $\{a, b\} \subseteq N \times M$).

Objects in $\mathcal{P}^{\mathsf{op}}\mathbf{Span}$ are sets, and a morphism $(l, r, n) : X \to Y$ consists of spans $X \xleftarrow{l} N \xrightarrow{r} Y$ in $\mathbf{Set}$ with a choice of subset of N, $n : I \to \mathcal{P}(N)$. The composition of $(l, r, n) : X \to Y$ (with apex N) and $(p, q, m) : Y \to Z$ (with apex M) is given by the apex $N \times_Y M = \{(n, m) \mid r(n) = p(m)\}$ of a span $X \leftarrow N \times_Y M \to Z$ and a subset $I \to \mathcal{P}(N \times_Y M)$ which we specify next. The legs of the pullback give the projections $\pi_N : N \times_Y M \to N$ and $\pi_M : N \times_Y M \to M$. Their pairing $\langle \pi_N, \pi_M \rangle : N \times_Y M \to N \times M$ maps pairs (n, m) that satisfy the equation $r(n) = p(m)$ to the same pairs,

only forgetting the constraint. $\mathcal{P}^{\mathsf{op}}\langle \pi_N, \pi_M \rangle : \mathcal{P}(N \times M) \to \mathcal{P}(N \times_Y M)$ precomposes a predicate $N \times M \to 2$ with the map $\langle \pi_N, \pi_M \rangle$. The identity on X in $\mathcal{P}^{\mathsf{op}}\mathbf{Span}$ is given by $X \leftarrow X \to X$ and $\mathrm{const}_\emptyset : I \to \mathcal{P}(X)$.

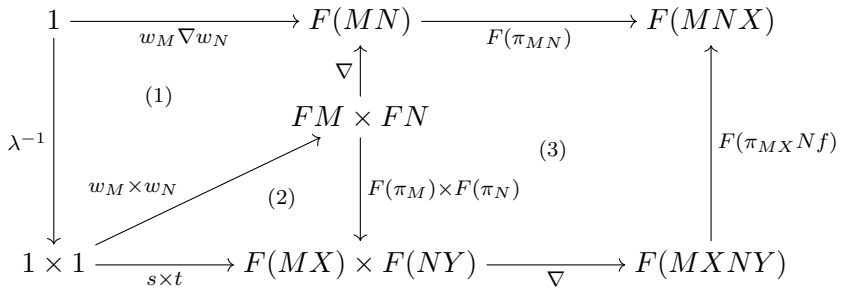
By simplifying the domain of application of both F-spans and weighted para, we can find a common specialization of both concepts.

Proposition 2.6.8. *For a cartesian monoidal $\mathcal{C}$ and a strong monoidal $F : \mathcal{C}^{\mathsf{op}} \to \mathbf{Set}$, there's a wide embedding functor $\iota : \mathbf{Para}_\times^{F^{\mathsf{op}}}(\mathcal{C}) \hookrightarrow F\mathbf{Span}$.*

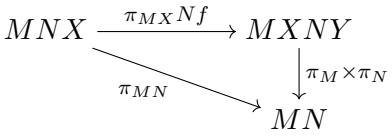
Proof. For a cartesian monoidal $\mathcal{C}$, consider $\mathbf{Para}_\times^{F^{\mathsf{op}}}(\mathcal{C})$ be as in example 2.6.2. ι is an identity on objects, and its action on morphisms is defined next. A morphism $X \to Y$ consists of $f : M \times X \to Y$ and $w \in FM$. Following proposition 2.5.6, this defines firstly a morphism $X \overset{\pi_X}{\leftarrow} M \times X \overset{f}{\to} Y$ in $\mathbf{Span}(\mathcal{C})$. Secondly, weights $w \in FM$ are bijective with generalized elements $w : 1 \to FM$, from which

$$s : 1 \overset{w}{\to} FM \xrightarrow{F(\pi_M)} F(M \times X) \tag{2.7}$$

gives structure elements from the monoidal unit $1 : \mathbf{Set}$. To prove functoriality for weights w_M, w_N, one has to check that their composition via (2.1) coincides with the composition of their corresponding structure elements s, t via (2.6). These respectively the top and bottom arrows in the following diagram, where we omit certain product symbols for brevity:



where (1) commutes by definition of the laxator and (2) by obtaining s and t from (2.7) applied to w_M and w_N respectively. (3) commutes by the image of the following diagram under F, noting that the laxator ∇ here is an isomorphism.



□

It remains to be explored whether the proposition holds for lax monoidal F and the faithfulness of ι. It should be noted however that $F\mathbf{Span}$ also includes morphisms whose structure elements are generalised elements $1 \to F(M \times X)$ that do not factor through FM, and as such do not arise from a weighted para construction. These will have their application in a second example of F-spans (example 4.2.18) that will motivate the technical machinery of chapter 4. After this, a concrete comparison with weighted para (definition 2.6.1) will be given in example 4.4.1.

Chapter 3

Dynamic programming

3.1 Introduction

Here 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, also known as discrete-time controlled Markov chains. The classical **Set**-based methods of computing optimal control policies, underlying some solution methods of both classical control theory and modern reinforcement learning, are known collectively as *dynamic programming*. These are based on two operations that can be interleaved in many ways: *value improvement* and *policy improvement*. The central idea of this chapter 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* $V : X \to \mathbb{R}$ into the reals 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 (definition 2.5.18). Every control policy π also induces an optic $\mathbb{B}(\pi) : \binom{X}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$. 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 $\binom{X}{\mathbb{R}} \xrightarrow{\mathbb{B}(\pi)} \binom{X}{\mathbb{R}} \xrightarrow{V} I$ is a closer approximation of the value of π. This is called *value improvement*. Iterating this operation

$$\dots \binom{X}{\mathbb{R}} \xrightarrow{\mathbb{B}(\pi)} \binom{X}{\mathbb{R}} \xrightarrow{\mathbb{B}(\pi)} \binom{X}{\mathbb{R}} \xrightarrow{V} I \tag{3.1}$$

converges efficiently to the true value function of the policy π under reasonable considerations[1] [133, §5.1].

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, again under reasonable assumptions[2] [133, §6.4.6].

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

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

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 [133, §6], [22, Ch.1].

In this chapter 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.

3.1.1 Related work

An earlier version of this chapter was written jointly with J. Hedges [85], who originally developed the idea of value iteration using open games around 2016 during discussions

[1] Because this is a limit in the analysis sense, the true value function exists when $\sup_{x \in X} \sup_{a \in A} |U(x,a)| < \infty$, where A is the action space and $U : X \times A \to \mathbb{R}$ is the utility function. See section 3.2.

[2] It is common in convex analysis to assume the codomain of a value function to be the *extended* reals, for which an argmax operation may have empty image. The policy improvement algorithm assumes that this does not happen, i.e. that there is always a maximizing decision rule at each $x \in X$.

with Viktor Winschel and Philipp Zahn [71], and planned also to be a section of [80] 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 [27] via optics [136]. In this chapter 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 [42], which largely subsumes open games [43]. Implementations of value iteration based on open games include a model from [15] of the social dilemma of emissions cuts and climate collapse as a stochastic game, or jointly controlled MDP by J. Hedges and Wolfram Barfuss [83] and an advanced implementation of reinforcement learning developed by Philipp Zahn, currently closed-source, used for the paper [56].

The most closely related work is [126], which formulates MDPs in terms of F-lenses (definition 2.3.6) of the functor $\mathrm{BiKl}(C \times -, \Delta(\mathbb{R} \times -))^{\mathsf{op}}$, where $C \times -$ is the reader comonad and $\Delta(\mathbb{R} \times -)$ is a probability monad over actions with their expected value. An MDP 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 in this chapter differs in two ways. We firstly assume that the readout function is the identity, as we are not dealing with partial observability [7]. Secondly, we discard the functoriality of the backwards update map with respect to the forward map, thus going from F-lenses to plain lenses, and specify a concrete structure of the backwards update map $f^* : X \times I \to \Delta(X \times \mathbb{R})$. This 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 chapter. Later, chapter 4 analyses how to recover the fibrational/functorial aspect of these bidirectional algorithms instantiated in the context of control theory.

Bakirtzis et al. propose a category of MDPs as models of tasks [14]. This emphasis on models allow them to compose different MDPs using fibre products and pushouts, and is agnostic to the control and RL algorithms that operate on them, which they take as given. Since our work focuses on a particular family of algorithms, we believe this approach is orthogonal to ours, and both could potentially be done simultaneously.

Another approach is to model MDPs as coalgebras [57, 149] from states to rewards and potential transitions, as done by Feys et al. [58]. They observe that the Bellman optimality condition for value iteration is a certain coalgebra-to-algebra morphism. We similarly believe this is orthogonal to our work and could potentially be unified.

When the state space is a convex polyhedron in a vector space and the value function is linear in the state, Bellman's equation is a linear programming problem [22]. A proof of Farkas' lemma in linear programming in the graphical calculus of Diagrammatic Polyhedral Algebra appears in [28], from Graphical Linear Algebra [32]. In this chapter we will not assume the state space to have any more structure than a set, and the setting of linear state spaces together with quadratic value functions will be considered in chapter 4.

A series of papers by Botta et al. (for example [34]) 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 partially tackle with open-loop systems also in chapter 4.

3.2 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 function $U : X \times A \to \mathbb{R}$ called utility or reward [133]. 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[3], who is trying to *control* the Markov chain with the objective of optimising the long-run reward.

A *policy*[4] 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

[3] Arguably, the use of 'agent' in this context is not appropriate, because the MDP and its transition probabilities and utility function are fully known, so there is no need to sample these functions. Chapter 5 will explore reinforcement learning agents, where only the MDPs interface X,A,$\mathbb{R}$ is known.

[4] We refer to policy, scheduler and control law interchangeably.

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 rewards for each transition:

$$V_\pi(x_0) = \sum_{k=0}^{\infty} \gamma^k U(x_k, \pi(x_k)) \tag{3.2}$$

where $0 < \gamma < 1$ 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.) This sum exists when $\sup_{x \in X} \sup_{a \in A} |U(x, a)| < \infty$. A key idea behind dynamic programming is that this geometric sum can be equivalently written as a telescoping sum:

$$V_\pi(x_0) = U(x_0, \pi(x_0)) + \gamma(U(x_1, \pi(x_1)) + \gamma(U(x_2, \pi(x_2) + \cdots))) \tag{3.3}$$

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 [19], we assume normative agents that choose the policy 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 [127].

3.3 Bellman operators

In dynamic programming, the agent's objective of maximizing the overall value 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[5].

[5]Several other denominations for these equations exist. In reinforcement learning, value and policy improvement are sometimes called Bellman expectation equation and Bellman optimality equation respectively. Value improvement is also called policy evaluation.

To define these update rules in eq. (3.4) and later, we use the tick V' to indicate the redefinition of V at x to be the RHS expression, keeping $V'(y) = V(y)$ for any other $y \neq x$. The type of these assignments is given by Bellman operators—mappings between function spaces of either state value functions or state-action value functions which iteratively improve the estimations.

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

$$V'(x) = U(x, \pi(x)) + \gamma V(f(x, \pi(x))) \tag{3.4}$$

  Because the update depends on both the previous value function V and a policy π, we write $\mathfrak{B}_{\text{val}}(V, \pi)(x) = U(x, \pi(x)) + \gamma V(f(x, \pi(x)))$ for the operator $\mathfrak{B}_{\text{val}} : \mathbb{R}^X \times A^X \to \mathbb{R}^X$.

- **Policy improvement** updates the policy function $\pi : X \to TA$ pointwise by traversing the state space X and updating the action taken in the state $\pi(x)$ with the action that maximises the discounted value after one step:

$$\pi'(x) = \operatorname*{argmax}_{a \in A} U(x, a) + \gamma V(f(x, a)) \tag{3.5}$$

  Similarly, we write $\mathfrak{B}_{\text{pol}}(V)(x) = \operatorname{argmax}_{a \in A} U(x, a) + \gamma V(f(x, a))$ for the operator $\mathfrak{B}_{\text{pol}} : \mathbb{R}^X \to A^X$.

A Bellman optimality condition on the other hand determines the fixpoint of these functional equations, and is met when $V' = V$ and $\pi' = \pi$ respectively. The solution V characterises the long-run values of the policy π. Provided X is finite and $0 < \gamma < 1$, $\mathfrak{B}_\pi$ is a contraction mapping on the supremum metric of $\mathbb{R}^X$, and therefore iterating $\mathfrak{B}_\pi$ from any initial estimate of V will converge to the unique solution of the Bellman equation in the limit [22].

The update rule (3.4) is the discounted sum (3.2) 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.

Depending on the sequencing of these two steps, we have three classic algorithms: Policy iteration iterates value improvement until the current policy value is optimal before performing a policy improvement step, value iteration interleaves both steps one after another, and generalized policy iteration interleaves m steps of policy improvement with n steps of value improvement.

Definition 3.3.1. *Policy iteration* (PIT) is an iterative algorithm in which an initial value function, typically initialized as $V(x) = 0$, is paired with a randomly selected policy π. The policy is evaluated by repeatedly applying the value improvement rule (3.4) until the value function converges to a fixed point $V' = V$, or until another convergence criterion is met. Convergence is guaranteed under standard conditions due to the contraction property of the Bellman operator [53]. Once convergence is achieved (or approximated), the policy is improved via the greedy update (3.5) yielding a new policy π' for the next iteration.

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

$$q_\pi(x, a) = U(x, a) + \gamma V(f(x, a)) \tag{3.6}$$

The *policy improvement theorem* [17] 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 steps

(3.4) and (3.5):

$$V_{\pi^*}(x) = \max_{a \in A} U(x, a) + \gamma V_{\pi^*}(f(x, a)) \tag{3.7}$$

Definition 3.3.2. *Value iteration* (VIT) is a variant of the policy iteration algorithm in which the value update step is limited to a single iteration, effectively truncating the policy evaluation sum (3.2) to its first summand. The value function is updated according to the assignment

$$V'(x) = \max_{a \in A} U(x, a) + \gamma V(f(x, a)) \tag{3.8}$$

which implicitly incorporates the value improvement step into the policy improvement. The corresponding policy at each iteration can be recovered by

$$\pi'(x) = \operatorname*{argmax}_{a \in A} U(x, a) + \gamma V(f(x, a)) \tag{3.9}$$

Definition 3.3.3. *Generalized Policy Iteration* (GPI) interleaves steps of policy evaluation and policy improvement in an iterative process, without requiring either step to fully traverse the state space X at each iteration. Rather than strictly alternating between exact policy evaluation and exact policy improvement, GPI allows partial or approximate updates of the value function and the policy.

At each iteration, the value function V is updated according to (3.4) applied to $n < |X|$ states, and the policy is then improved via (3.5) at $m < |X|$ possibly different states, with respect to the updated value function. This algorithm generalizes both value iteration and policy iteration as special cases.

GPI converges by the same contraction argument, as the operations here are pointwise applications of the same Bellman operators.

3.3.1 Stochastic dynamic programming

Stochasticity can be introduced in different places in an MDP.

1. in the policy $\pi : X \to TA$ for a probability monad T on **Set**, 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 $f : X \times A \to \mathrm{D}X$ and potentially the reward function $U : X \times A \to \mathrm{D}\mathbb{R}$ independently.

3. usually the reward is included inside the transition function $f : X \times A \to \mathrm{D}(X \times \mathbb{R})$, 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 $f : X \times A \to \mathrm{D}X \times \mathrm{D}\mathbb{R}$.

Here we make a conceptual distinction between the monad T—which can be chosen by the agent, as it can choose to play a deterministic policy, or a stochastic policy, or a non-deterministic policy, etc.—and the monad D, which is given in the statement of the MDP. For the sake of exposition, in the remaining sections of this chapter we will assume D and T both to be 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 (3.4) becomes stochastic, and adopts a slightly different form depending on which part of the MDP is stochastic. For the cases 1. and 2.:

$$V'(x) = \sum_a \pi(a \mid x)(U(x,a) + \gamma V(f(x,a)))$$
$$V'(x) = \sum_r U(r \mid x,a)r + \sum_{x'} f(x' \mid x,a)\gamma V(x')$$

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

$$\begin{aligned} V'(x) &= \mathbb{E}_{\substack{a\sim\pi(x) \\ (x',r)\sim f(x,a)}} [r + \gamma V(x')] \\ &= \sum_{a\in A} \pi(a \mid x) \sum_{x',r} f(x', r \mid x, a)(r + \gamma V(x')) \end{aligned} \tag{3.10}$$

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 operators type becomes $\mathfrak{B}_{\mathrm{val}} : \mathbb{R}^X \times (TA)^X \to \mathbb{R}^X$. Given a fixed policy π, we write the shorthand

$\mathfrak{B}_\pi = \mathfrak{B}_{\text{val}}(-,\pi) : \mathbb{R}^X \to \mathbb{R}^X$, which has the expression

$$\mathfrak{B}_\pi(V)(x) = \mathbb{E}_{(r,x')\sim t(x,\pi(x))}[r + \gamma V(x')] \tag{3.11}$$

Likewise, the policy improvement rule (3.5) becomes:

$$\pi'(x) = \operatorname*{argmax}_a \mathbb{E}_{(x',r)\sim t(x,a)}[r + \gamma V(x')] \tag{3.12}$$

whose operator has type $\mathfrak{B}_{\text{pol}} : \mathbb{R}^X \to (TA)^X$.

The policy improvement theorem holds in the stochastic setting too [158, §4.2] by defining

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

Notation 3.3.4. The operator expressions allow a compact notation of the algorithms seen so far. Let $(-)^\dagger$ be the (in practice approximate) fixpoint of the operator. Given $\mathfrak{B}_{\text{pol}} : \mathbb{R}^X \to (TA)^X$ (3.12) and $\mathfrak{B}_{\text{val}} : \mathbb{R}^X \times (TA)^X \to \mathbb{R}^X$ (3.10), let $\overline{\mathfrak{B}_{\text{pol}}} = (\Delta_{\mathbb{R}^X} \times !_{(TA)^X}) \fatsemi (1_{\mathbb{R}^X} \times \mathfrak{B}_{\text{pol}})$ and $\overline{\mathfrak{B}_{\text{pol}}} = (1_{\mathbb{R}^X} \times \Delta_{(TA)^X}) \fatsemi (\mathfrak{B}_{\text{val}} \times 1_{(TA)^X})$ be the maps $(TA)^X \times \mathbb{R}^X \to (TA)^X \times \mathbb{R}^X$ associated to each of them (fig. 3.1).

Figure 3.1: $\overline{\mathfrak{B}_{\text{val}}}$ (left) and $\overline{\mathfrak{B}_{\text{pol}}}$ (right) in the cartesian monoidal category **Set**.

We can summarise the three dynamic programming algorithms as:

Policy iteration:	$(\overline{\mathfrak{B}_{\text{val}}}^\dagger \fatsemi \overline{\mathfrak{B}_{\text{pol}}})^\dagger$
Value iteration:	$(\overline{\mathfrak{B}_{\text{val}}} \fatsemi \overline{\mathfrak{B}_{\text{pol}}})^\dagger$
Generalized policy iteration:	$(\overline{\mathfrak{B}_{\text{val}}}^m \fatsemi \overline{\mathfrak{B}_{\text{pol}}}^n)^\dagger$ for some $m, n \leq \lvert S \rvert$

3.3.2 Gridworld

A classic example from 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. By assuming perfect knowledge of the finite MDP, we can use the DP algorithms introduced so far. If the MDP's transition function or reward functions were not known, the agent's learning algorithm needs samples obtained by interacting with the environment, which we treat in chapter 5.

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 fig. 3.2).

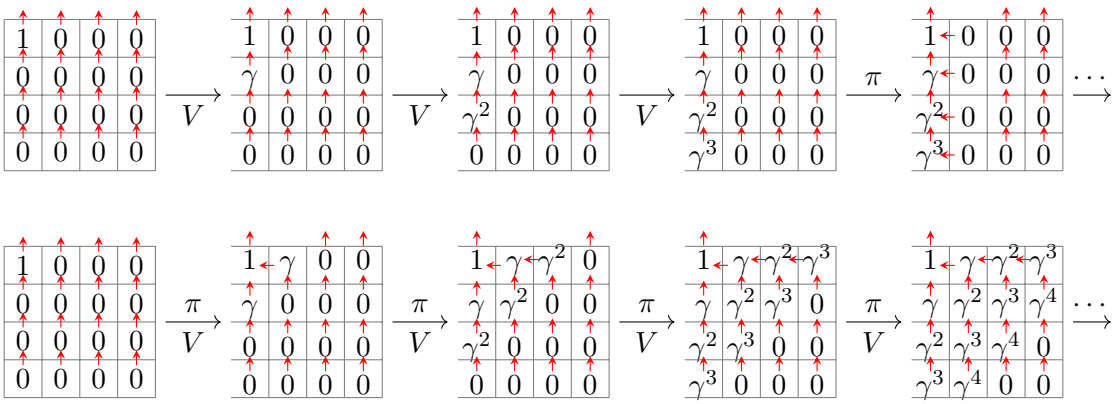


Figure 3.2: Policy iteration (above) and value iteration (below). The numbers in the cells are state values and the red arrows are the directions dictated by the policy at each stage. The arrows between grids indicate what kind of update the algorithm does, either value improvement (V) or policy improvement (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 and converges exactly after three iterations.

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. Later in chapter 4, a characterisation of dependent action spaces will be shown in example 4.3.3.

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 $1 - \epsilon$ 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.

3.3.3 Inverted pendulum

A task that illustrates a continuous state space MDP is the control of a pendulum balanced over a cart, a ubiquitous problem used to illustrate ideas in the control systems literature (see e.g. [8]). The dynamics can be described in continuous-time by two non-linear differential equations [152][66, Example 2E]:

$$\begin{aligned}(\bar{m} + m)\ddot{x} + mL\ddot{\theta}\cos\theta - mL\dot{\theta}^2\sin\theta &= u \\ mL\ddot{x}\cos\theta + mL^2\ddot{\theta} - mLg\sin\theta &= 0\end{aligned} \tag{3.13}$$

where the constants are the mass of the cart $\bar{m}$, the mass of the pendulum m, the length of the pendulum L and the gravitational constant g (fig. 3.3). The variables are the angle of the pendulum with respect to the upwards position θ, the carts horizontal position x and our control u. The state variables define a point $\boldsymbol{x} = (x, \dot{x}, \theta, \dot{\theta})$ in the (smooth) state manifold[6] $M = \mathbb{R} \times \mathbb{R} \times \mathbb{S}^1 \times \mathbb{R}$, where horizontal position x, linear velocity $\dot{x}$ and angular velocity $\dot{\theta}$ take unconstrained real values, while the angle takes values in the circle $\mathbb{S}^1$.

The non-linear differential equations (3.13) define a map $f : M \times \mathbb{R} \to TM$ from

[6] A state manifold is the state space of all configurations of the system. Because of the inherent non-linearity that some variables like angles have, this set has the structure not of a vector space but of a manifold, i.e. a topological space that locally looks like $\mathbb{R}^n$ (in our case, like $\mathbb{R}^4$).

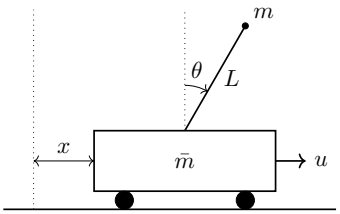


Figure 3.3: Inverted pendulum on a moving cart.

the state manifold M and the control space $\mathbb{R}$ to tangents[7] in M. For a fixed control u, $f_u : M \to TM$ is a vector field, i.e. a section of the tangent bundle $TM \to M$. The cost associated to the state and control is given by a map $\ell : M \times U \to \mathbb{R}$.

Let $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, u)$ refer to the continuous-time control-affine system [33]

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = f(\boldsymbol{x}(t), u(t)) \tag{3.14}$$

Sampling a trajectory by $\boldsymbol{x}_k = \boldsymbol{x}(k\Delta t)$ and fixing a control law $\pi : X \to U$, one can define the discrete-time propagator $\Phi = \phi_{\Delta t}$ by

$$\phi_{\Delta t}(\boldsymbol{x}(t)) = \boldsymbol{x}(t) + \int_t^{t+\Delta t} f(\boldsymbol{x}(\tau))\mathrm{d}\tau \tag{3.15}$$

which allows to model the system with $\boldsymbol{x}_{k+1} = \Phi(\boldsymbol{x}_k)$, where we take the state space as the global M and the action space of the force exerted to the cart as $U = \mathbb{R}$.

A further simplification of the problem is to observe that the system of equations (3.14) can be *linearized* by Taylor near a state $\boldsymbol{x}_0$ and control u_0 as

$$\dot{\boldsymbol{x}} \approx f(\boldsymbol{x}_0, u_0) + A_{\boldsymbol{x}_0}(\boldsymbol{x} - \boldsymbol{x}_0) + B_{u_0}(u - u_0) \tag{3.16}$$

where the Jacobian of f defines $A_{\boldsymbol{x}_0,u_0} = \partial_{\boldsymbol{x}} f|_{\boldsymbol{x}_0,u_0}$ and $B_{\boldsymbol{x}_0,u_0} = \partial_u f|_{\boldsymbol{x}_0,u_0}$. This linearization makes the dynamics a *linear* vector field on the tangent space $T_{\boldsymbol{x}_0}M$.

When the state $\boldsymbol{x}_0$ is a (not necessarily stable) equilibrium point, like the pendulum

[7] The tangent bundle $TM = \sum_{p\in M} T_pM$ is the disjoint union of all tangent spaces T_pM. The tangent space T_pM at a point p consists of all tangent vectors $\gamma'(0)$ to smooth curves γ on M that pass through p.

being in the upwards position, we can assume the small-angle approximations[8] $\cos\theta \approx 1$ and $\sin\theta \approx \theta$, as well as small velocities leading to negligible quadratic terms $\dot{\theta}^2 \approx 0$ and $\dot{x}^2 \approx 0$, which allow for A and B to be constants given by

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -mg/\bar{m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & (\bar{m}+m)g/(\bar{m}L) & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1/\bar{m} \\ 0 \\ -1/\bar{m}L \end{bmatrix}$$

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 $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta t(A\boldsymbol{x}_k + Bu_k)$, with the same constants, where k indexes time steps. Additionally, the immediate cost associated to the pair $(\boldsymbol{x}_k, u_k)$ will be given by the cost accumulated in the Δt interval $\tilde{\ell}(\boldsymbol{x}_k, u_k) := \int_{t_k}^{t_k+\Delta t} \ell(\boldsymbol{x}(t_k), u(t_k))\mathrm{d}t$. 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 $K : X \to U$ can be learned. Under linearisation near an equilibrium, we take the state space as $X = T_{\boldsymbol{x}_0}M = \mathbb{R}^4$. The action space does not depend on the linearisation and always be taken as $U = \mathbb{R}$.

The time-discretised formulation of this problem is more common in reinforcement learning settings than in control theory. In that case, 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.

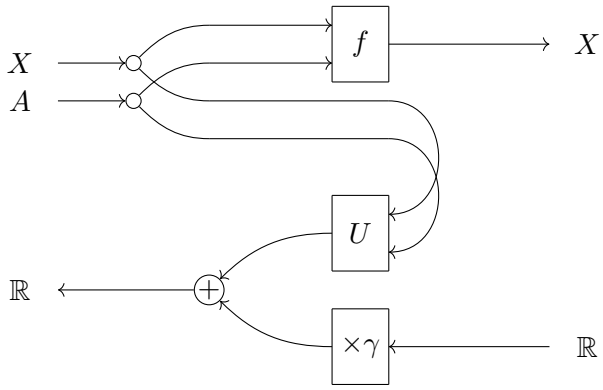
$$J(\boldsymbol{x}, u) = \int_0^\infty \ell(\boldsymbol{x}(t), u(t))\mathrm{d}t$$

[8] In calculus, the small-angle approximation is the first-order approximation of the Taylor expansion of trigonometric functions $\sin\theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \cdots$ and $\cos\theta = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$.

3.4 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 specific category of optics in which this lives can be tailored to some extent, and depends on how much typing information we choose to include.

The definition of this optic is given by the following string diagram (recall fig. 2.2):



(3.17)

To be clear, this diagram is not completely formal if instantiated in an arbitrary mixed optic category $\mathbf{Optic}_{\mathcal{M}}(\mathcal{C}, D)$. In general, we require the forwards category $\mathcal{C}$ to be a Markov category (giving us copy morphisms (2.2) Δ_X and Δ_A), and the backwards category $\mathcal{D}$ must have a suitable object[9] $\mathbb{R}$, together with morphisms $\times\gamma : \mathbb{R} \to \mathbb{R}$ and $+ : \mathbb{R} \otimes \mathbb{R} \to \mathbb{R}$.

A formal interpretation exists for example when the forwards category acts on the backwards category (recall section 2.5.1), i.e. $\mathcal{C} = \mathcal{M} \to \mathcal{D}$. Then the forwards pass is a morphism $g : X \otimes A \to X \otimes X \otimes A$ in $\mathcal{C}$ where

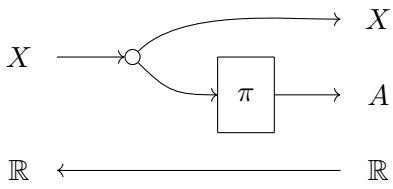
$$g = \Delta_{X \otimes A} \fatsemi (f \otimes \mathrm{id}_{X \otimes A})$$

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

$$\psi = (X \otimes A, g, g') : \binom{X \otimes A}{\mathbb{R}} \to \binom{X}{\mathbb{R}} \tag{3.18}$$

[9] We denote it by $\mathbb{R}$ as we assume in this chapter to be working with the semiring of reals with regular addition and multiplication and the natural order. The properties of this object will be studied further in chapter 4.

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



(3.19)

Here we are also assuming that the forwards category has 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

$$(I, \Delta_X \fatsemi (\mathrm{id}_X \otimes \pi), \mathrm{id}_{\mathbb{R}}) \tag{3.20}$$

Notice that the domain (resp. codomain) of ψ (3.17) coincides with the codomain (resp. domain) of $\overline{\pi}$ (3.19). This allows for an alternating sequential composition of the two. Given a fixed policy π, we refer to the composition of the transition optic after the policy optic as the 'value improvement optic' $\mathbb{B}(\pi)$:

$$\mathbb{B}(\pi) := \overline{\pi} \fatsemi \psi : \binom{X}{\mathbb{R}} \to \binom{X}{\mathbb{R}} \tag{3.21}$$

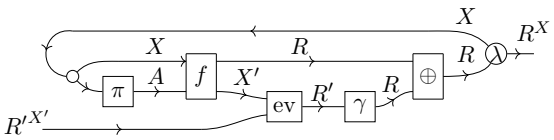
Proposition 3.4.1. *The Bellman operator* $\mathfrak{B}_\pi$ (3.11) *is recovered as* $\mathfrak{B}_\pi = \mathbb{K}(\mathbb{B}(\pi))$, *where* $\mathbb{K}$ *is the continuation functor* (2.5).

Proof. Let a value function $V : X \to \mathbb{R}$ be represented by a costate of optics $V : \binom{X}{\mathbb{R}} \to \binom{1}{1}$. Then the costate $\mathbb{B}(\pi) \fatsemi V : \binom{X}{\mathbb{R}} \to \binom{1}{1}$ represents $\mathfrak{B}_\pi(V) : X \to \mathbb{R}$. □

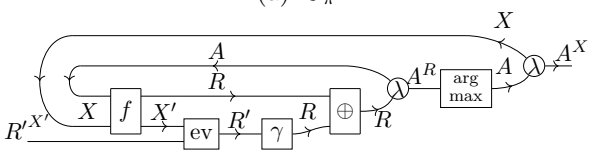
The above proposition is a refinement of the usual view of Bellman operators as endomorphisms of function spaces. However, as we see next, not all Bellman operators are images under $\mathbb{K}$ of some optic. In particular, even though both $\mathfrak{B}_{\mathrm{val}}$ and $\mathfrak{B}_{\mathrm{pol}}$ are treated as similar contraction operators in the dynamic programming literature, the policy improvement step (3.23) cannot be stated as an optic.

Proposition 3.4.2. *Let* $\mathfrak{B}_{\mathrm{pol}} : \mathbb{R}^X \to A^X$ *be the policy improvement Bellman operator* (3.5). *There is no optic* $(M, g, g') : \binom{X}{A} \to \binom{X}{\mathbb{R}}$ *such that* $\mathfrak{B}_{\mathrm{pol}} = \mathbb{K}(M, g, g')$.

Proof. Recall from fig. 2.4 the image of an optic under $\mathbb{K}$. $\mathfrak{B}_{\text{pol}}$ involves two applications of the currying operation (denoted λ), and therefore does not lie in the image of an optic under the continuation functor (fig. 3.4). □



(a) $\mathfrak{B}_\pi$



(b) $\mathfrak{B}_{\text{pol}}$

Figure 3.4: Bellman operators in the free autonomisation of **Set** [52] (see also remark 2.5.19).

3.5 Applications

3.5.1 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 $\gamma \in (0, 1)$. We convert this into an optic $\psi = (X \times A, g, g') : \binom{X \times A}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$ in **Optic**(**Set**), whose forwards pass is $g(x, a) = (x, a, f(x, a))$ and whose backwards pass is $g'(x, a, r) = U(x, a) + \gamma r$.

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:

$$\textbf{Value improvement:} \qquad (\pi, V) \mapsto (\pi, \mathbb{B}(\pi) \fatsemi V) \tag{3.22}$$

$$\textbf{Policy improvement:} \qquad (\pi, V) \mapsto (x \mapsto \operatorname*{argmax}_{a \in A} (\psi \fatsemi V)(x, a), V) \tag{3.23}$$

We assume that argmax 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, we can see that (3.22) does correspond to (3.8), as it replaces V with

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

and likewise (3.23) corresponds to (3.9) by replacing π with

$$\pi'(x) = \operatorname*{argmax}_{a \in A} U(x, a) + \gamma V(f(x, a))$$

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

$$V^*(x) = \max_{a \in A}(\psi \fatsemi V^*)(x, a) = \max_{a \in A} U(x, a) + \gamma V^*(f(x, a))$$
$$\pi^*(x) = \operatorname*{argmax}_{a \in A}(\psi \fatsemi V^*)(x, a) = \operatorname*{argmax}_{a \in A} U(x, a) + \gamma V^*(f(x, a))$$

Example 3.5.1 (Gridworld example)**.** A policy of an agent in our version of Gridworld (fig. 3.2) 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 $\mathbb{B}(\pi) \fatsemi V$, which differs with V only at $(0, 1) \mapsto \gamma$.

If we instead perform a policy improvement step, the value function remains unchanged while the new policy differs with π at $(1, 0) \mapsto \operatorname{argmax}_{a \in A}(\psi \fatsemi V)(1, 0, a) = \leftarrow$.

3.5.2 Continuous-space deterministic decision processes

The optic associated to the inverted pendulum on a cart problem presented in section 3.3.3 can be defined for the non-linear (smooth) dynamics as well as the linearisation around an equilibrium point. We revisit the linearisation around arbitrary states in the next chapter (example 4.4.10), and present the smooth version next.

Example 3.5.2 (Inverted pendulum)**.** A state of our time-discretized inverted pendulum on a cart consists of $\boldsymbol{x} = (x, \dot{x}, \theta, \dot{\theta})$ in the state space $X = \mathbb{R}\times\mathbb{R}\times\mathbb{S}^1\times\mathbb{R}$. The non-linear discrete-time propagator $\Phi : x_i \mapsto x_{i+1}$ is a smooth map $X \to X$.

The cost $J(\boldsymbol{x}, u) = \sum_{i=0}^{\infty} \gamma^i \tilde{\ell}(\boldsymbol{x}_i, u_i)$ defines the backwards smooth function $X \times U \times \mathbb{R} \to \mathbb{R}$ which adds the cost at the i-th time step $\tilde{\ell}(\boldsymbol{x}_i, u_i)$ to the discounted sum:

$$\left(\boldsymbol{x}_i, u_i, \sum_{j=i+1} \gamma^j \tilde{\ell}(\boldsymbol{x}_j, u_j)\right) \mapsto \sum_{j=i} \gamma^j \tilde{\ell}(\boldsymbol{x}_j, u_j)$$

These two maps form an optic $\psi : \binom{X\times U}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$ in $\mathbf{Optic}(\mathbf{Mfd}_\infty)$. In conclusion, the two optics involved in this example are

$$\begin{aligned}
\psi = \binom{\Phi}{J} : \binom{X\times A}{\mathbb{R}} \to \binom{X}{\mathbb{R}} \qquad & \overline{\pi} = \binom{\pi}{p_2} : \binom{X}{\mathbb{R}} \to \binom{X\times U}{\mathbb{R}} \\
\Phi : X \times U \to X \qquad & \mathrm{gr}(\pi) : X \to X \times U \\
(\boldsymbol{x}, u) \mapsto \phi_{\Delta t}(\boldsymbol{x}, u) \qquad & \boldsymbol{x} \mapsto (\boldsymbol{x}, \pi(\boldsymbol{x})) \\
J : X \times U \times \mathbb{R} \to \mathbb{R} \qquad & p_2 : X \times \mathbb{R} \to \mathbb{R} \\
(\boldsymbol{x}, u, r) \mapsto \tilde{\ell}(\boldsymbol{x}, u) + \gamma r \qquad & (_, r) \mapsto r
\end{aligned}$$

Remark 3.5.3. Note that the cost function $\tilde{\ell}$ 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). Quadratic forms are a common example of convex functions; they will be discussed in more detail in section 4.2, with further justification for their prominence provided in remark 4.2.14.

Remark 3.5.4. In numerical implementations, a naive formalisation of continuous state spaces $X = \mathbb{R}^n$, $U = \mathbb{R}^m$ makes the exhaustive search of a sequence of k optimal

controls have complexity $\mathcal{O}(kM^k)$ for $M = |\mathbb{R}^m|$ the cardinality of the set of points in the vector space $\mathbb{R}^m$. The dynamic programming approach has complexity $\mathcal{O}(kNM)$ for $N = |\mathbb{R}^n|$, which although better than exhaustive search, is still exponential in n, m and thus prohibitive; $\mathbb{R}$ is not enumerable. In the machine learning community, one of the interpretations of the *curse of dimensionality* refers to this phenomenon: the number of samples required to estimate a function accurately grows exponentially with n [26, §1.4].

Common approaches to this problem include the quantisation of the state space into a possibly non-uniform grid, where X is a distribution over the barycentric coordinates of a simplicial complex obtained e.g. by triangulation, and linearisation, which we mentioned at the beginning of the example and develop in chapter 4. Other approximation methods include splines, hierarchical algorithms, memory-based methods, and the restriction of the space of values to a family of parametrized functions [141, §4]. In the context of reinforcement learning, we discuss gradient-based methods in chapter 5 (section 5.2.3).

3.5.3 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 $\gamma \in (0, 1)$. We can write the transition function as conditional probabilities $f(x' \mid x, a)$.

We can convert this data into an optic $\psi : \binom{X \otimes A}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$ 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 $g'(\alpha)(r) = \mathbb{E}U(\alpha) + \gamma r$, 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 $X \otimes A$ (although it cannot be considered free on an object of **Mark**), and define it as the linear extension of $g'(x, a)(r) = U(x, a) + \gamma r$.

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

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

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:

$$V^*_{\pi^*}(x_0) = \mathbb{E} \sum_{i=0}^{\infty} \gamma^i U(x_i, \pi^*(x_i))$$

Example 3.5.5 (Gridworld, continued)**.** In a proper MDP, transition functions can be stochastic, and update steps have to take expectations over values: value improvement maps $(\pi, V) \mapsto (\pi, \mathbb{B}(\pi) \fatsemi V)$ and policy improvement maps $(\pi, V) \mapsto (x \mapsto \operatorname{argmax}_{a \in A} \mathbb{E}(\psi \fatsemi V)(x,a), V)$. 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 [98, §2], a problem which is known in control theory as the identification-control conflict.

3.6 Towards Q-learning

Consider a deterministic decision process corresponding to the optic $\psi : \binom{X \times A}{\mathbb{R}} \to \binom{X}{\mathbb{R}}$. 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 (3.6) rather than a state-value function $V : X \to \mathbb{R}$.

We can define similar update steps

$$\begin{aligned} &\textbf{Value improvement:} && (\pi, q) \mapsto (\pi, \psi \fatsemi \overline{\pi} \fatsemi q) \\ &\textbf{Policy improvement:} && (\pi, q) \mapsto \left(x \mapsto \operatorname*{argmax}_{a \in A} q(x,a), q \right) \end{aligned}$$

These can also be fused into a single step:

$$\textbf{State-action value iteration:} \qquad (\pi, q) \mapsto \left(x \mapsto \operatorname*{argmax}_{a \in A} q(x,a), \psi \fatsemi \overline{\pi} \fatsemi q \right) \tag{3.24}$$

Observe that the only difference with the regular value iteration (definition 3.3.2) seen in section 3.5.1 is the flipped order of composition of the ψ optic with $\overline{\pi}$, 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 to represent a Bellman operator $\mathbb{R}^{X\times A} \to \mathbb{R}^{X\times A}$. Other than that, the convergence properties of both are the same, as starting with a state value function v and performing value iteration steps following (3.8) is equivalent to starting with a state-action value function $q(x,a) = v(x)$ and updating by (3.24). So one might wonder, what is the point of learning state-action value functions $X \times A \to \mathbb{R}$ rather than state-value functions $X \to \mathbb{R}$?

The advantage of this variation is that it gives a way to approximate $\operatorname{argmax}_{a\in A}(\psi \fatsemi \overline{\pi} \fatsemi q)(x,a)$ *without making any use of* ψ, namely by instead using $\operatorname{argmax}_{a\in A} q(x,a)$. This leads to an effective method known as Q-learning [166] 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 dynamic programming and *reinforcement learning*. The above method, despite learning a q-function, is *not* Q-learning because it makes use of ψ during value improvement. This will be the topic of chapter 5, where we explore Q-learning and other RL algorithms in the context of agent-environment interaction.

3.7 Other research avenues

In addition to the implementation of several families of MDPs illustrated in section 3.5, the category theoretic perspective enables the practical development of a software library or embedded language for combinators for compositional MDP solutions, allowing engineers to synthesize optimal control strategies for large-scale systems by linking verified or pre-solved modules using the bidirectional control flow strategies provided by the data-accessor interpretation of lenses. Furthermore, the string diagrammatic syntax can be leveraged to represent these processes formally and aid the implementation process in particular domains of knowledge.

Chapter 4

Control Theory

4.1 Control for the category theorist

In engineering, control theory gives the mathematical foundation for the prediction and regulation of the behaviour of dynamical systems in order to achieve certain outcomes, often using feedback. Two cornerstones of modern control are the Linear Quadratic Regulator (LQR) and the Kalman filter (KF), which represent control problems (e.g. how to best navigate a physical environment using minimal energy) and estimation problems (e.g. how to infer the state of a dynamical system from noisy observations) respectively under concrete enough assumptions and conditions to allow an optimal solution that is closed-form (see [4] for a detailed exposition).

Firstly, an assumption for both LQRs and KFs is that the dynamics of the system or 'plant' in question are linear, i.e. that the state and control spaces have vector space structure and that the next state is a linear function of the current state and control input. Secondly, LQRs constraint the control law to be linear, and assume that the cost functions that are to be minimized are quadratic functions of the state and control, which penalize state deviations (i.e. how far the system is from its target) and control effort (i.e. how much energy is used to correct the system). KFs assume that the noise corruption of the observations are zero-mean Gaussian.

The celebrated Kalman duality [100] shows that these two problems are solved by

the same equation, namely the (matrix) Riccati equation[1]: by assigning the variables of an LQR problem to the Riccati equation, one obtains a map from states to controls that minimises the overall energy. Conversely, assigning the variables of a KF problem *to the same Riccati equation* determines the map from measurements to state estimations that minimises the mean square error.

The Riccati equation connects moreover with our work so far, by being (informally) the 'linear-algebra version' of the Bellman equation, which was presented as an optic in chapter 3. Being at the heart of some algorithms used to solve LQRs, KFs and other control problems, one might ask whether there is also a bidirectional composition of these algorithms. This chapter is aimed at answering this question. It is tempting to propose a candidate solution (if the Bellman equation is a morphism in **Lens**(**Set**), surely the Riccati equation is a morphism in **Lens**($\mathbf{FVec}_{\mathbb{R}}$)!), but analysing why these don't work is often very hard without engaging with the existing literature on how these algorithms came to be in the first place.

We thus take as a guiding example first the problem of computing optimal controls for LQRs. Along our efforts to give a categorical account of the computation of these optimal controls, we will see how developing the necessary notation leads us naturally to certain (modest) results on time-varying systems. Specifically, we aim to show how the Riccati equation emerges as a functorial construction in a category of linear systems, and how the control-estimation duality is a functor that relates LQRs and Kalman filters. Understanding LQRs as 'analytical MDPs', the value improvement and policy improvement steps are going to be characterised here as constraint pullback and Schur complements.

Our theoretical tools will centre around Grothendieck constructions, which we introduced in section 2.3, and optimality as an order relation and quadratic forms, which we will present throughout this chapter.

[1]The matrix version of the original ordinary differential equation by J.Riccati [134] was mostly tied to the development of LQRs and KFs themselves.

4.1.1 Related work

In the intersection of control theory and category theory, [61] gives a structural characterisation of controllability, and [55] gives a comprehensive account of controllability and observability of systems in a category of signal flow diagrams [30, 31, 29], a widely used tool in control theory. Compared to their time-agnostic and (therefore inevitably) time-invariant formalisation, our approach is motivated by two factors: (*i*) transient regime behaviour, and (*ii*) closeness to the implementation of algorithms. This results in an inherently discrete time and time-variant theory that couples the discretization of time with a morphism's source and target. Referring back to the argument in section 1.1, composition in their categories is 'relational' whereas ours is 'directional'—their morphisms are (open) systems whereas ours are processes. We compare their treatment of Kalman duality with ours in section 4.4.4.

Linear quadratic regulators are a particular case of model predictive control (MPC), which has been formalized in a category of parametrised convex bifunctions [76]. Their work focuses around the compositional *specification* of the problems—in a similar vein to how we build $\mathrm{Sym}_{+}^{\mathsf{op}}$ **Span**—whose solution is then calculated by an external solver. Our aim in this chapter however is to prove that the *solutions* to these control problems are also compositional, by restricting to a case—linear dynamics, quadratic costs—where the convexity of cost functions has an algebraic characterisation and the minimization has an expression (the Riccati equation) which we are able to define as a functor to a category of *optimal* solutions.

4.2 Quadratic forms over a vector space

For a set X, $\mathcal{P}(X)$ is the set of all subsets of X, which forms a complete lattice under the inclusion order. Focusing on the join semilattice structure of $\mathcal{P}(X)$ with the empty set $\emptyset$ as least element, let $\mathcal{P} : \mathbf{FinSet} \to \mathbf{Pos}$ be the covariant powerset functor into the category of posets and monotone functions.

Elements of $\mathcal{P}(X)$ are also functions $X \to 2$, that is, predicates on X. As $\mathcal{P}$ is covariant, given a function $f : X \to Y$, $\mathcal{P}(f)$ maps $P : 1 \to \mathcal{P}(X)$ to a predicate

$Q : 1 \to \mathcal{P}(Y)$, defined as $Q(y) = \exists x. f(x) = y \wedge P(x)$.

We introduce next a similar construction for vector spaces and linear maps, where instead of subsets (predicates) we will have quadratic forms (cost functions). For V a finite-dimensional vector space, the set $\mathrm{Sym}(V)$ of quadratic forms contains functions $Q : V \to \mathbb{R}$ such that for all $\boldsymbol{u}, \boldsymbol{v} \in V$ and all $\alpha \in \mathbb{R}$, $Q(\alpha \boldsymbol{v}) = \alpha^2 Q(\boldsymbol{v})$ and $Q(\boldsymbol{u} + \boldsymbol{v}) - Q(\boldsymbol{u}) - Q(\boldsymbol{v})$ is bilinear. An element $Q \in \mathrm{Sym}(V)$ for $V \cong \mathbb{R}^n$ is determined by $n(n+1)/2$ values in $\mathbb{R}$ corresponding to the upper diagonal entries of the matrix (4.1), since the lower diagonal ones are determined by the corresponding transpose.

$$Q = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{12}^\top & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{1n}^\top & q_{2n}^\top & \cdots & q_{nn} \end{bmatrix} \tag{4.1}$$

A functorial definition of this assignment is given next.

Definition 4.2.1. The *symmetric square functor* $\mathrm{Sym} : \mathbf{FVec}_\mathbb{R} \to \mathbf{FVec}_\mathbb{R}$ maps objects $n : \mathbf{FVec}_\mathbb{R}$ to $\frac{n(n+1)}{2}$, which correspond to the number of independent components that define a quadratic form, i.e. a symmetric bilinear form. On morphisms $f : n \to m$, $\mathrm{Sym}(f) = f(-)f^\top$ is a linear map $\mathrm{Sym}(n) \to \mathrm{Sym}(m)$ given by pre-composition with the transpose[2] and post-composition with f. $\mathrm{Sym}(f)$ preserves the vector space structure on $\mathbb{R}^n$ that is defined by pointwise addition and scalar multiplication.

Similarly, a contravariant functor $\mathrm{Sym}^{\mathsf{op}} : \mathbf{FVec}_\mathbb{R}^{\mathsf{op}} \to \mathbf{FVec}_\mathbb{R}$ is defined on morphisms as $\mathrm{Sym}^{\mathsf{op}}(f) = f^\top(-)f : \mathrm{Sym}(m) \to \mathrm{Sym}(n)$.

Remark 4.2.2. This 'free' definition of Sym is equivalent to its 'concrete' definition: all elements of the set $\mathrm{Sym}(V)$, i.e. symmetric bilinear forms Q over $V \cong \mathbb{R}^n$, are given by generalized elements $1 \to \mathrm{Sym}(n)$ by means of the diagonal form $Q = \sum_{i=1}^n \lambda_i q_i q_i^\top$.

We emphasize that the image of a morphism under Sym is linear, because composition of linear maps is linear. Sym is different from the endomorphism functor

[2] We assume a dagger structure on $\mathbf{FVec}_k$ in the form of transpose $(-)^\top$ for $k = \mathbb{R}$ or complex conjugation $(-)^*$ for $k = \mathbb{C}$.

End : **FVec** $\to$ **FVec** defined in 'concrete terms' on objects as $V \mapsto V \otimes V^*$ and on morphisms as sending linear $h : V \to W$ to linear $h \otimes h^* : V \otimes V^* \to W \otimes W^*$, as the map on objects in free terms is $n \mapsto n^2$. We illustrate the difference between Sym, Sym^{op} and End in the next example.

Example 4.2.3. Consider $f : 3 \to 2$ given by $f(x, y, z) = (2x, y)$, or as a matrix $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ under suitable choice of bases for $\mathbb{R}^3$ and $\mathbb{R}^2$. The image of this morphism under Sym, Sym^{op} and End is shown next:

- The map $\text{Sym}(f) = f(-)f^\top : \text{Sym}(3) \to \text{Sym}(2)$ is given by pre-composition with f and post-composition with its transpose:

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{12}^\top & v_{22} & v_{23} \\ v_{13}^\top & v_{23}^\top & v_{33} \end{bmatrix} \mapsto \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{12}^\top & v_{22} & v_{23} \\ v_{13}^\top & v_{23}^\top & v_{33} \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 4v_{11} & 2v_{12} \\ 2v_{12}^\top & v_{22} \end{bmatrix}$$

- The map $\text{Sym}^{\text{op}}(f) = f^\top(-)f : \text{Sym}(2) \to \text{Sym}(3)$ is given by pre-composition with the transpose and post-composition with f:

$$\begin{bmatrix} v_{11} & v_{12} \\ v_{12}^\top & v_{22} \end{bmatrix} \mapsto \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} \\ v_{12}^\top & v_{22} \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 4v_{11} & 2v_{12} & 0 \\ 2v_{12}^\top & v_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- The domain (resp. codomain) of $\text{End}(f) = f \otimes f^*$ are 3^2 (resp 2^2), which are bigger than $\text{Sym}(3) = 6$ and $\text{Sym}(2) = 3$.

Notice that $\text{Sym}(1) = 1$. By remark 4.2.2, to pick a particular quadratic form out of $\text{Sym}(n)$ means to define a morphism $\text{Sym}(1) \to \text{Sym}(n)$. Keeping track of these pointed objects leads to the following definition.

Definition 4.2.4. Let $\text{Sym}_\bullet := \text{Hom}(\text{Sym}(1), \text{Sym}(-)) : \textbf{FVec} \to \textbf{FVec}$ be a functor that maps n to the vector space of all quadratic forms over $\mathbb{R}^n$, which we denote with a $\bullet$ subscript to denote these pointed objects. Let $\text{Sym}_\bullet^{\text{op}} : \textbf{FVec}^{\text{op}} \to \textbf{FVec}$ be is

contravariant analogue. The category of elements $\mathbf{El}(\mathrm{Sym}_\bullet^{\mathsf{op}})$ (recall section 2.3.1) has as objects pairs (n, Q) of a vector space dimension n and a quadratic form Q in the fibre over n. A morphism $(n, Q) \to (m, J)$ is a linear map $f : \mathbb{R}^n \to \mathbb{R}^m$ such that $f^\top J f = Q$.

Proposition 4.2.5. Sym *is a lax monoidal functor* $(\mathbf{FVec}, +, 0) \to (\mathbf{FVec}, +, 0)$.

Proof. The laxator $\nabla : \mathrm{Sym}(n) + \mathrm{Sym}(m) \to \mathrm{Sym}(n+m)$ is given by 'embedding into a block diagonal matrix':

$$\begin{aligned} & \mathrm{Sym}(n) + \mathrm{Sym}(m) \\ \xrightarrow{\lambda^{-1}} \; & \mathrm{Sym}(n) + 0 + 0 + \mathrm{Sym}(m) \\ \xrightarrow{\mathrm{id} \oplus ! \oplus ! \oplus \mathrm{id}} \; & \mathrm{Sym}(n) + (n \times m^\top) + (n^\top \times m) + \mathrm{Sym}(m) \\ \cong \; & \mathrm{Sym}(n+m) \end{aligned}$$

□

Quadratic forms are classified by Sylvester's law of inertia in terms of the number of positive, zero and negative eigenvalues. Those with no negative eigenvalues are called *positive semi-definite* forms (the 'positive cone'), and those with no positive eigenvalues are called *negative semi-definite* forms (the 'negative cone').

Remark 4.2.6 (The partial order on $\mathrm{Sym}(n)$)**.** $\mathrm{Sym}(n)$ has a partial order where for $X, Y : 1 \to \mathrm{Sym}(n)$, $X \leq Y$ if $Y - X$ is positive semi-definite (*Löwner order*) [25], i.e. $\forall \boldsymbol{v} \in \mathbb{R}^n . \boldsymbol{v}^\top (Y - X) \boldsymbol{v} \geq 0$. Although other orders can be defined on $\mathrm{Sym}(n)$ (like spectral order, i.e. the majorization of their eigenvalues in $\mathbb{R}_+$, which is weaker [118, §9.L]), Our focus is on the Löwner order because it is compatible with its vector space structure: it's translation invariant ($X + Z \leq Y + Z$) and respects positive scalar multiplication (for $c \in \mathbb{R}_+$ and $X \leq Y$, $c \cdot X \leq c \cdot Y$). In contrast, spectral order for example is not translation invariant, as eigenvalues do not behave linearly under arbitrary matrix addition: $\lambda_i(X + Y) \neq \lambda_i(X) + \lambda_i(Y)$ (see Weyl's inequality, e.g. in [25, §III.2]).

Proposition 4.2.7. *The Löwner order on* $\mathrm{Sym}(n)$ *is total if and only if* $n \in \{0, 1\}$. *It does not have least upper bounds.*

Proof. The order on $\mathrm{Sym}(0)$ is trivial, while $\mathrm{Sym}(1) = 1$ inherits the natural order of positive reals. For $\mathrm{Sym}(2)$, two elements $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1/2 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} 1/2 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ are incomparable. The identity matrix I is an upper bound for which there's no strictly smaller upper bound, whereas $B = \frac{1}{2}\left(\begin{smallmatrix} 2.8 & 1 \\ 1 & 2.8 \end{smallmatrix}\right)$ is another upper bound, yet B and I are incomparable. Thus, the set $\left\{\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1/2 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1/2 & 0 \\ 0 & 1 \end{smallmatrix}\right)\right\}$ does not have a least upper bound in $\mathrm{Sym}(\mathbb{R}^2)$. Similar counterexamples can be constructed for any dimension higher. □

Example 4.2.8. The non-existence of least upper bounds between $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1/2 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} 1/2 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ can be illustrated in the context of control. Consider a vector space $\mathbb{R}^2$ whose two dimensions encode the state and the control spaces. The two square matrices encode cost functions, and the partiality of the Löwner order means that these two cost functions cannot be compared.

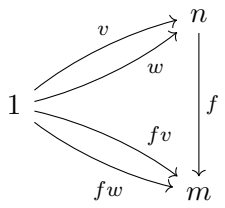
The Löwner order defines a poset structure on $\mathrm{Hom}(\mathrm{Sym}(1), \mathrm{Sym}(n))$, and by enriching over **Pos**, the functor $\mathrm{Sym}_\bullet : \mathbf{FVec} \to \mathbf{Pos}$ (cf. definition 4.2.4) assigns the Löwner order on the homsets. Note that in definition 4.2.4, witnessing the vector space structure on quadratic forms led to define $\mathrm{Sym}_\bullet : \mathbf{FVec} \to \mathbf{FVec}$, while witnessing the order structure on them changes the codomain to **Pos**. It turns out that for the Löwner order, one can accommodate both the linear structure and the order structure, and we can define $\mathrm{Sym}_\bullet : \mathbf{FVec} \to \mathbf{FVec}^{\leq} \to \mathbf{Cat}$ to the category $\mathbf{FVec}^{\leq}$ defined below, as shown later in proposition 4.2.10.

Definition 4.2.9. Let $\mathbf{FVec}_k^{\leq}$ be the concrete category of partially ordered finite-dim. k-vectors and monotone[3] linear maps.

For a linear map $f : n \to m$, consider the following diagram of two vectors in $\mathbb{R}^n$

[3] We use *monotone* as in order-preserving (that is, preserving the Löwner order), not in the sense of operator monotone functions [114].

and their corresponding image in $\mathbb{R}^m$.



(4.2)

The poset structure makes $\mathrm{Sym}_\bullet(n)$ a thin category, in addition to its vector space structure. Moreover, $\mathrm{Sym}_\bullet$ sends linear maps to monotone morphisms in $\mathbf{FVec}^{\leq}$ that preserve positive semi-definiteness, thus defining a functor into **Cat**.

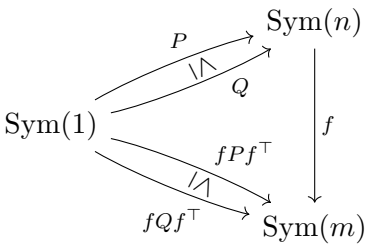
Proposition 4.2.10. *The functor* $\mathrm{Sym}_\bullet : \mathbf{FVec} \to \mathbf{Cat}$ *is well-defined.*

Proof. The homset $\mathrm{Sym}_\bullet(n) = \mathrm{Hom}(\mathrm{Sym}(1), \mathrm{Sym}(n))$ of quadratic forms on $n : \mathbf{FVec}$ forms a thin category where $P \to Q$ is non-empty iff $P \leq Q$. $\mathrm{Sym}_\bullet(f)$ preserves positive semi-definiteness: Assume that an object $P : 1 \to \mathrm{Sym}(n)$ is positive semi-definite, i.e. it satisfies $\forall \boldsymbol{v} \in \mathbb{R}^n . \boldsymbol{v}^\top P \boldsymbol{v} \geq 0$. Let $\mathrm{Sym}_\bullet(f)(P) = Q$. For any $w \in \mathbb{R}^m$ we can choose $v \in \mathbb{R}^n$ such that $v = f^\top w$, and

$$\boldsymbol{w}^\top Q \boldsymbol{w} = \boldsymbol{w}^\top (fPf^\top)\boldsymbol{w} = (f^\top \boldsymbol{w})^\top P (f^\top \boldsymbol{w}) = \boldsymbol{v}^\top P \boldsymbol{v} \geq 0$$

By linearity $\mathrm{Sym}_\bullet(f)$ preserves Löwner order too: given $P \leq Q$, $fPf^\top \leq fQf^\top$. This defines a functor $\mathrm{Sym}_\bullet(n) \to \mathrm{Sym}_\bullet(m)$. □

The following diagram is the image of (4.2) under Sym, or alternatively, the image of $f : n \to m$ under $\mathrm{Sym}_\bullet$.



By a similar proof, $\mathrm{Sym}_\bullet^{\mathsf{op}}$ is a functor $\mathbf{FVec}^{\mathsf{op}} \to \mathbf{Cat}$ that maps $f : n \to m$ to $f^\top(-)f : \mathrm{Sym}_\bullet(m) \to \mathrm{Sym}_\bullet(n)$.

Restricting the functor $\mathrm{Sym}_\bullet$ to either the positive or negative cones weakens the algebraic structure on the codomain to semimodule elements (elements of 'modules over semirings').

Proposition 4.2.11. *Let $\mathbb{R}_+$ be the semiring of positive reals, and $\mathbf{SMod}^{\leq}_{\mathbb{R}_+}$ the concrete category of partially ordered semimodule elements over $\mathbb{R}_+$ and monotone semimodule homomorphisms. The mappings of a vector space to its positive semi-definite (resp. negative semi-definite) forms define lax monoidal functors*

$$\mathrm{Sym}_{\bullet+}, \mathrm{Sym}_{\bullet-} : (\mathbf{FVec}, +, 0) \to (\mathbf{SMod}^{\leq}_{\mathbb{R}_+}, +, 0) \tag{4.3}$$

Proof. Let F be either $\mathrm{Sym}_{\bullet+}$ or $\mathrm{Sym}_{\bullet-}$. $F(n)$ has an additive commutative monoid structure and scalar multiplication $\times : \mathbb{R}_+ \times F(n) \to F(n)$ inherited from the vector space structure. Let $f : n \to m$ be a linear map, $X, Y \in F(n)$ two quadratic forms, and $c \in \mathbb{R}_+$. $Ff : F(n) \to F(m)$ preserves addition and scalar multiplication by

$$\begin{aligned} Ff(X) + Ff(Y) &= fXf^\top + fYf^\top = f(X+Y)f^\top = (Ff)(X+Y) \\ c \cdot Ff(X) &= c \cdot (fXf^\top) = f(cX)f^\top \end{aligned}$$

Order is preserved as multiplication by an element of $\mathbb{R}_+$ is monotone, and lax monoidality is proved by the same argument as proposition 4.2.5. $F(n)$ does not have a vector space structure over $\mathbb{R}$ as the multiplication of a positive (resp. negative) semidefinite form by a negative scalar is not positive (resp. negative) semidefinite. □

The above defined functors $\mathrm{Sym}_{\bullet+}$ and $\mathrm{Sym}^{\mathsf{op}}_{\bullet+}$ are 'quantitative versions' of $\mathcal{P}$ and $\mathcal{P}^{\mathsf{op}}$, in the sense that the functorial mapping of sets to their subsets embeds into the functorial mapping of vector spaces to their positive semi-definite forms. Remark 4.2.15 will show why both the vector space structure and the partial order are necessary on $\mathrm{Sym}_{\bullet+}$ for our applications.

Definition 4.2.12. Let the functor $F = \mathrm{Sym}^{\mathsf{op}}_{\bullet+} : \mathbf{FVec}^{\mathsf{op}} \to \mathbf{Cat}$ be defined as in proposition 4.2.11. The categories $\mathbf{Lens}_{\mathrm{Sym}^{\mathsf{op}}_{\bullet+}}$, $\mathbf{Chart}_{\mathrm{Sym}^{\mathsf{op}}_{\bullet+}}$, $\mathbf{Lens}_{\mathrm{Sym}_{\bullet+}}$ and $\mathbf{Chart}_{\mathrm{Sym}_{\bullet+}}$

are defined analogously to their category of elements (definition 4.2.4) on objects. A morphism $(g, \leq) : (n, Q) \to (m, J)$ in each of the four categories are as follows:

- In $\mathbf{Lens}_{\mathrm{Sym}_{\bullet+}^{\mathrm{op}}}$, a linear map $g : \mathbb{R}^n \to \mathbb{R}^m$ such that $g^\top J g \leq Q$.
- In $\mathbf{Chart}_{\mathrm{Sym}_{\bullet+}^{\mathrm{op}}}$ is a linear map $g : \mathbb{R}^n \to \mathbb{R}^m$ such that $Q \leq g^\top J g$.
- In $\mathbf{Lens}_{\mathrm{Sym}_{\bullet+}}$, a linear map $g : \mathbb{R}^n \to \mathbb{R}^m$ such that $J \leq g Q g^\top$.
- In $\mathbf{Chart}_{\mathrm{Sym}_{\bullet+}}$ is a linear map $g : \mathbb{R}^n \to \mathbb{R}^m$ such that $g Q g^\top \leq J$.

Proposition 4.2.13. *The four categories in definition 4.2.12 are well-defined.* $\mathbf{Lens}_{\mathrm{Sym}_{\bullet+}^{\mathrm{op}}} \to \mathbf{FVec}$ *and* $\mathbf{Chart}_{\mathrm{Sym}_{\bullet+}^{\mathrm{op}}} \to \mathbf{FVec}$ *are fibrations, and* $\mathbf{Lens}_{\mathrm{Sym}_{\bullet+}} \to \mathbf{FVec}$ *and* $\mathbf{Chart}_{\mathrm{Sym}_{\bullet+}} \to \mathbf{FVec}$ *are opfibrations.*

Proof. Let us first show that $\mathbf{Lens}_{\mathrm{Sym}_{\bullet+}^{\mathrm{op}}}$ has cartesian morphisms for every $g : n \to m$ in $\mathbf{FVec}$ and $(m, J \in \mathrm{Sym}_{\bullet+}(m))$. The preimage of (m, J) along g is the set $g^{-1}(m, J) := \{(n, Q \in \mathrm{Sym}_{\bullet+}(n)) \mid g^\top J g \leq Q\}$, which retains the preorder structure of $\mathrm{Sym}_{\bullet+}(n)$. The pullback of (m, J) along g, which we will denote $g^\sharp(m, J)$, has a unique (vertical) morphism from all elements of $g^{-1}(m, J)$. A vertical morphism $(n, Q) \to g^\sharp(m, J)$ (from any element $(n, Q) \in g^{-1}(m, J)$ of the preimage to the pullback) in $\mathbf{Lens}_{\mathrm{Sym}_{\bullet+}^{\mathrm{op}}}$ is given by $\tilde{Q} \leq Q$ (note the contravariance), so the pullback is the least element of $g^{-1}(m, J)$. Note that even though Löwner order does not have least upper bounds of arbitrary subsets (proposition 4.2.7), the minimum operation is taken over elements that are above $g^\top J g$, and is thus the unique $g^\top J g$ itself.

To show that $(g, \leq) : (n, Q') \to (m, J)$ where $Q' = \min g^{-1}(m, J) = g^\top J g$ is indeed cartesian, consider any morphism $(gf, \leq) : (l, P) \to (m, J)$. This is given by

$$P \leq (gf)^\top J (gf) = f^\top (g^\top J g) f = f^\top Q' f$$

The category $\mathrm{Sym}_{\bullet+}(l)$ is thin—a preorder—so the morphism $(f, \leq) : (l, P) \to (n, Q')$ given by $f : \mathbb{R}^l \to \mathbb{R}^n$ such that $P \leq f^\top Q' f$ is unique.

$\mathbf{Chart}_{\mathrm{Sym}_{\bullet+}^{\mathrm{op}}}$ similarly has cartesian morphisms, where pullbacks are the greatest upper bounds of $g^{-1}(m, J) := \{(n, Q \in \mathrm{Sym}_{\bullet+}(n)) \mid Q \leq g^\top J g\}$, again given by $(n, g^\top J g)$.

The proofs for $\mathbf{Lens}_{\mathrm{Sym}_{\bullet+}} \to \mathbf{FVec}$ and $\mathbf{Chart}_{\mathrm{Sym}_{\bullet+}} \to \mathbf{FVec}$ are similar. □

Note that injectivity of g is not necessary for $\mathbf{Lens}_{\mathrm{Sym}^{\mathsf{op}}_{\bullet+}}$ and $\mathbf{Chart}_{\mathrm{Sym}^{\mathsf{op}}_{\bullet+}}$ to be well-defined. This is possible because we define the maps in the fibres to consist of order inequalities and not e.g. linear maps (semimodule homomorphisms) in $\mathrm{Sym}_{\bullet+}(n)$. We illustrate this difference specifically for F-lenses by comparing with dependent lenses (F-lenses for $\mathrm{Slice} : \mathcal{C}^{\mathsf{op}} \to \mathbf{Cat}$ [151, Example 3.5]) next. A morphism $(f, f^\sharp) : (c, p : x \to c) \to (d, q : y \to d)$ in $\mathbf{Lens}_{\mathrm{Slice}}$ consists of $f : c \to d$ in $\mathcal{C}$ and $f^\sharp : f^*(y) \to x$ in $\mathcal{C}/c$. $f^\sharp$ makes the following triangle commute by definition.

$$\begin{array}{ccc} c \times_d y & \xrightarrow{\quad f^\sharp \quad} & x \\ & \searrow \quad \swarrow & \\ & c & \end{array} \tag{4.4}$$

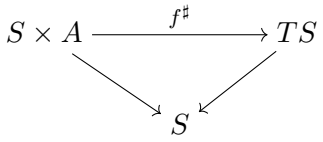
In contrast, a morphism $(f, \leq) : (n, Q) \to (m, J)$ in $\mathbf{Lens}_{\mathrm{Sym}^{\mathsf{op}}_{\bullet+}}$ is given by $f : n \to m$ in $\mathbf{FVec}$ and $\leq\colon f^\top J f \to Q$ in $\mathrm{Sym}_{\bullet+}(n)$ such that the following diagram 'commutes'.

$$\begin{array}{ccc} f^\top J f & \xrightarrow{\quad \leq \quad} & Q \\ & \searrow \quad \swarrow & \\ & n & \end{array} \tag{4.5}$$

If maps in $\mathrm{Sym}_{\bullet+}$ were linear, the putative morphism $f^\sharp : f^\top J f \to Q$ requires f to be injective. It should be noted however that while eq. (4.4) is a proper diagram in $\mathcal{C}$, eq. (4.5) is not a diagram in a particular category; the horizontal arrow is a morphism in $\mathrm{Sym}_{\bullet+}(n)$ while the arrows pointing at n are the object components of the forgetful functor to $\mathbf{FVec}$.

The dissimilarity with the specialization of dependent lenses for *open dynamical systems* [160, 126], [151, Example 3.6] follows a similar argument. By recognizing the natural manifold structure on $\mathrm{Sym}_{\bullet+}(n)$ (as an open subset of the vector space $\mathrm{Sym}(\mathbb{R}^n)$ with its euclidean topology and dimension $n(n+1)/2$), one might consider $\mathrm{Sym}_{\bullet+} : \mathbf{FVec}^{\mathsf{op}} \to \mathbf{Cat}$ to be a specialization of $\mathrm{Subm} : \mathbf{Mfd}^{\mathsf{op}}_{\geq 1} \to \mathbf{Cat}$. However, a morphism $(f, f^\sharp) : (S, TS) \to (B, A \times B)$ in $\mathbf{Lens}_{\mathrm{Subm}}$ consists of a smooth $f : S \to B$ and $f^\sharp : f^*(A \times B) \to TS$ where the pullback $f^*(A \times B)$ simplifies to $S \times A$, such that

the following triangle commutes:



This triangle can again be embedded in $\mathbf{Mfd}_{\geq 1}$, as $f^\sharp$ is a morphism between submersions so in particular a differentiable map, and the submersion maps (pointing to S) too.

Remark 4.2.14. The use of quadratic functions for costs—as opposed to general convex functions for example— is motivated by several reasons. Firstly, the matrix expression the Hessian gives an algebraic characterisation of convexity via positive semi-definiteness. Secondly, they allow an explicit solution method for several control and optimization problems based on the fact that first order optimality conditions are *linear* with respect to state and control variables. Last but not least, even in cases where cost functions are given by more involved smooth functions, the appearance in their second-order Taylor expansion serves as a 'canonical' local approximation of smooth functions. This modelling choice is consistent with standard practice in the literature [36, 130].

4.2.1 Constraints as Lagrangians and pullbacks

Let (X, P_X) and (Y, P_Y) be two pairs of state spaces (vector spaces) and cost functions (positive semi-definite forms over the spaces), and $f : X \to Y$ a linear map. A single step of the **delayed rewards** or **accumulated costs problem**—specialized for linear dynamics and quadratic costs—is the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{x} \in X}{\operatorname{argmin}} \quad & J(\boldsymbol{x}) = \boldsymbol{x}^\top P_X \boldsymbol{x} + \boldsymbol{y}^\top P_Y \boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{y} = f(\boldsymbol{x}) \end{aligned} \tag{4.6}$$

where the aggregated cost quantity J is another positive semi-definite form over X called the *cost-to-go* or *cumulative cost* is calculated from the immediate costs P_X and

P_Y. The objective is to find a certain $\boldsymbol{x} \in X$ that minimizes the cumulative cost while respecting the constraint $\boldsymbol{y} = f(\boldsymbol{x})$ imposed by the dynamics, and we informally refer to the construction of such a cost-minimizing state as the *optimal estimator*.

We review first the Lagrangian approach used in linear algebra to solve this problem, which can be found in any standard textbook [36, 74][130, Ch.17], and then we will give an alternative 'categorical' solution method via $\mathbf{Lens}_{\mathrm{Sym}_+^{\mathrm{op}}}$, whose morphisms can be thought of as selecting feasible, possibly non-optimal solutions (upper sets).

Lagrangian approach The minimization of J with respect to the constraints coincides with the unconstrained minimization of its *Lagrangian relaxation*, i.e. a cost function $\mathcal{L}$ defined as

$$\mathcal{L}(x, \lambda, y) = J(x, y) + \lambda^\top (Ax - y) \tag{4.7}$$

where we write A instead of f for the linear map. Because $\mathcal{L}$ is still a symmetric positive semi-definite quadratic form (over $X \oplus \mathbb{R} \oplus Y$, where $\mathbb{R}$ is the domain of the *Lagrange multiplier*), its minimum is characterized by the first-order necessary condition $\nabla\mathcal{L}(x, \lambda, y) = 0$. Each of the components defines an equation of the linear system whose solution is the optimal (x, λ, y):

$$\begin{aligned} \partial_x \mathcal{L}(x, \lambda, y) &= 0 \\ \partial_\lambda \mathcal{L}(x, \lambda, y) &= 0 \\ \partial_y \mathcal{L}(x, \lambda, y) &= 0 \end{aligned} \qquad \begin{bmatrix} P_X & A^\top & \\ A & 0 & -I \\ & -I & P_Y \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

The argmin operator is in general multi-valued, which in the context of vector spaces means that it picks a subspace of X, and only in certain well-behaved situations picks a single value. Because $\mathcal{L}$ is quadratic, $\nabla\mathcal{L}$ is linearly dependent on (x, λ, y) and can be expressed itself as a matrix as shown above. This allows to express the solution subspace of $X \oplus \mathbb{R} \oplus Y$ as

$$\operatorname*{argmin}_{(x,\lambda,y)} \{J(x) \mid y = f(x)\} = \ker(\nabla\mathcal{L})$$

One may observe two things from this toy example: By the banded structure of

$\nabla\mathcal{L}$, the computation of $\ker(\nabla\mathcal{L})$ via gaussian elimination requires limited pivoting operations. Secondly, determining the optimal subspace of X requires computing the optimal subspaces of $\mathbb{R}$ and Y. We address these two points in the next approach.

Pullback approach The pairs (X, P_X) and (Y, P_Y) define two objects (n, P_n) and (m, P_m) of $\mathbf{Lens}_{\mathrm{Sym}_{\bullet+}^{\mathsf{op}}}$ with $n = \dim X$ and $m = \dim Y$. The cartesian lift of $f : n \to m$ determines $(n, f^\top P_m f)$. This object and (n, P_n) are in the same fibre $\mathrm{Sym}_{\bullet+}(n)$, whose linear structure allows computing $(n, P_n + f^\top P_m f)$. The kernel of $P_n + f^\top P_m f$ is the optimal subspace of $\mathbb{R}^n$. Moreover, this circumvents the differentiation of $\mathcal{L}$ and the introduction of Lagrange multipliers in the gaussian elimination algorithm.

$$\pi_n \circ \ker(\nabla\mathcal{L}) = \ker(P_n + f^\top P_m f) \tag{4.8}$$

Remark 4.2.15. It should be noted that both the partial order defining morphisms and semimodule structure are necessary on $\mathrm{Sym}_{\bullet+}(n)$ because the pointwise addition $P+Q$ does not satisfy the universal property of coproducts (recall the non-existence of least upper bounds from proposition 4.2.7). The pointwise addition on the right-hand side of (4.8) cannot be expressed when only considering these cost matrices as elements of a poset.

Theorem 4.2.16. $\mathbf{El}(\mathrm{Sym}_{\bullet+}^{\mathsf{op}})$ *is the optimal estimator of quadratic costs over linear systems.*

Proof. The fibration $\int \mathrm{Sym}_{\bullet+}^{\mathsf{op}} \to \mathbf{FVec}$ has a single cleavage, which given a linear map $f : n \to m$ and a cost $P \in \mathrm{Sym}_{\bullet+}(m)$ picks $f^\top P f$ over n. The identity-on-objects embedding $\iota : \mathbf{El}(\mathrm{Sym}_{\bullet+}^{\mathsf{op}}) \hookrightarrow \int \mathrm{Sym}_{\bullet+}^{\mathsf{op}}$ also picks $f^\top P f$ in the fibre, which corresponds to the minimum cost-to-go. □

When considering accumulated costs, we would like to apply the same 'Grothendieck construction' to a base category of uncontrolled processes. For $P_n + f^\top P_m f$ in (4.8) to be the image of a morphism in $\mathbf{SMod}_{\mathbb{R}_+}^{\leqslant}$, note that the expression is linear in P_n and P_m *separately.*

Example 4.2.17. Consider a discrete linear dynamical process $f : \mathbb{R}^n \to \mathbb{R}^m$ embedded in $\mathbf{LinSpan}_{\mathbb{R}}$ (proposition 2.5.8) as $n = n \xrightarrow{f} m$. Given costs $P \in \mathrm{Sym}_{\bullet+}(n)$ and $Q \in \mathrm{Sym}_{\bullet+}(m)$ associated to the current and future state spaces, the optimal cost-to-go at n is given by the morphism $f^\top P f + Q \leq J$ over the apex.

$$\begin{array}{ccc} & & J \\ & & \leq \uparrow \\ \int \mathrm{Sym}_+^{\mathsf{op}} & Q \longrightarrow f^\top P f + Q & \longleftarrow P \\ {\scriptstyle p}\downarrow & & \\ \mathbf{FVec}_{\mathbb{R}} & n = n \xrightarrow[f]{} & m \end{array}$$

4.2.2 Linear spans and cospans decorated by quadratic costs

The category **FVec** and the functor Sym are special, insofar that the biproduct structure on **FVec** and the covariance of $\mathrm{Sym}_{\bullet+} : \mathbf{FVec} \to \mathbf{SMod}^{\leq}_{\mathbb{R}_+}$ (resp. contravariance of $\mathrm{Sym}^{\mathsf{op}}_{\bullet+} : \mathbf{FVec}^{\mathsf{op}} \to \mathbf{SMod}^{\leq}_{\mathbb{R}_+}$) allows to define both decorated cospans (definition 2.6.4) and decorated spans (definition 2.6.5) for them. We give an explicit construction for their decorated span category in example 4.2.18.

Monoids (resp. comonoids) in the monoidal category $(\mathbf{FVec}_k, +, 0)$ are k-algebras (definition A.2.5) (resp. k-coalgebras). The canonical example of a structure-indexing comonoid in $\mathbf{FVec}_{\mathbb{R}}$ is the $\mathbb{R}$-algebra $(1, \Delta)$, where $\Delta(*) = (*, *)$. Similarly, a natural choice for a structure-indexing comonoid in $\mathbf{SMod}^{\leq}_{\mathbb{R}_+}$ is the base semiring $\mathbb{R}_+$ of positive reals (i.e. the object 1) with the same Δ. Having the definition of $\mathcal{P}^{\mathsf{op}}\mathbf{Span}$ (example 2.6.7) in mind, we define the category of decorated spans for the functor $\mathrm{Sym}^{\mathsf{op}}_+$ next.

Example 4.2.18. The category $\mathrm{Sym}^{\mathsf{op}}_+ \mathbf{Span}$ of spans decorated by $\mathrm{Sym}^{\mathsf{op}}_+$ consists of spans of linear maps $n \xleftarrow{f} u \xrightarrow{g} m$ and structure elements $\mathrm{Sym}_+(1) \cong \mathbb{R}_+ \to \mathrm{Sym}_+(u)$ being a choice of positive semi-definite quadratic form over u.

Identity is given by $n \xleftarrow{\mathrm{id}} n \xrightarrow{\mathrm{id}} n$ and zero quadratic form $0_n : \mathbb{R}_+ \to \mathrm{Sym}_+(n)$.

Remark 4.2.19. Discounting cannot be accounted for in the structure-indexing comonoid, as the map $\boldsymbol{x} \mapsto (\boldsymbol{x}, \gamma\boldsymbol{x})$ is not coassociative $((\boldsymbol{x}, \gamma(\boldsymbol{x}, \gamma\boldsymbol{x})) : \mathbb{R} \oplus (\mathbb{R} \oplus \mathbb{R}) \neq ((\boldsymbol{x}, \gamma\boldsymbol{x}), \gamma\boldsymbol{x}) :$

$(\mathbb{R} \oplus \mathbb{R}) \oplus \mathbb{R})$ nor counital.

4.2.3 Markov categories for the location-scale family

Definition 4.2.20. Let **Gauss** be the Markov category of Euclidean spaces and affine functions with gaussian noise [67, §6]. That is, a morphism $(f, \Sigma, \mu) : n \to m$ in **Gauss** consists of a linear map $f : \mathbb{R}^n \to \mathbb{R}^m$, a covariance $\Sigma : \mathrm{Sym}_+(\mathbb{R}^m)$ and a mean $\mu : \mathbb{R}^m$ which define the conditional distribution $Y : D\mathbb{R}^m$ with respect to $X : D\mathbb{R}^n$ as

$$Y = fX + \xi$$

with $\mathbb{E}[\xi] = \mu$ and $\mathrm{Var}[\xi] = \Sigma$.

Gauss is an example of a Markov category, and is conjectured [67] not to arise as a Kleisli category of a monad. We will assume that euclidean spaces carry appropriate σ-algebras when taking measures on them. Recall that for a lax monoidal functor $F : \mathcal{C} \to \mathbf{Set}$, $\mathbf{El}(F)$ is monoidal [145, 123]. For $\mathrm{Hom}(I, -) : \mathbf{Gauss} \to \mathbf{Set}$, the category $\mathbf{El}(\mathrm{Hom}(I, -)) \cong \mathbf{Gauss}_\bullet$ of pointed gaussian distributions attaches to each euclidean space $\mathbb{R}^n$ an associated distribution as a generalized element $(0, C, s) : I \to n$.

There is a strong monoidal functor $\mathbf{Gauss} \to \mathbf{Euc}_{\mathrm{Aff}}$ to the category of euclidean spaces and affine linear functions [67]. We will explore how this functor relates to the opfibration $\mathbf{Chart}_{\mathrm{Sym}_+} \to \mathbf{FVec}_{\mathbb{R}}$ (proposition 4.2.13), and explore similar opfibrations for a wider family of distributions that are closed under pushforward of measures along affine transformations. We start by looking at their definition for the single-variable case.

Definition 4.2.21 (cf. [44, §3.5])**.** Let $f : D\mathbb{R}$ be a probability distribution over the reals. For any *location parameter* $-\infty < \mu < \infty$ and any *scale parameter* $\sigma > 0$, the univariate *location-scale family* is the family of probability distributions generated by

$$\frac{1}{\sigma} f\left(\frac{\boldsymbol{x} - \mu}{\sigma}\right) \tag{4.9}$$

The location-scale family generated from many common distributions is functionally

closed, i.e. their parametrisation retains the functional form. This means that we can characterise these distributions by their parameters, and that the distribution (4.9) can be interpreted as a conditional distribution with respect to a standard distribution, and we may type $f : 1 \to 1$ in an appropriate category $\mathcal{LS}_{\mathcal{D}}$ of Euclidean spaces and certain probability kernels to be specified. The location-scale family then defines the homset $\mathcal{LS}_{\mathcal{D}}(1, 1)$.

For this category to be well-defined, morphisms should parametrise Markov kernels whose Chapman-Kolmogorov composition must be in the family, or in other words, the family needs to be closed under *compound* distributions (in the sense of mixture models, not compound Poisson distributions). Our focus is on the location-scale family, rather than other families encompassing gaussian kernels (such as the exponential family). This restriction is motivated by the requirement that, for the associated category to be opfibered over $\mathbf{Euc}_{\mathrm{Aff}}$, distributions have to be represented by (hyper)parameters that transform under affine maps in a compatible way. Some examples in the location-scale family are univariate gaussians, uniform distributions and Student's t-distributions.

Example 4.2.22 (Uniform distributions over a closed interval)**.** The standard *uniform distribution* over $\mathbb{R}$ is given by the probability density function

$$f(\boldsymbol{x}) = \begin{cases} \frac{1}{2} & -1 \leq \boldsymbol{x} \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

The family $\mathcal{LS}_{\mathrm{Unif}}(1, 1)$ consists of all uniform distributions over $\mathbb{R}$, characterised by the mean and width $(\mu, w) : \mathbb{R} \times \mathbb{R}_+$, given by

$$f(\boldsymbol{x}) = \begin{cases} \frac{1}{2w} & \mu - w \leq \boldsymbol{x} \leq \mu + w \\ 0 & \text{otherwise} \end{cases}$$

Example 4.2.23 (Student's t-distributions)**.** Let ν be a fixed positive integer. The

standard *Student's t-distribution* over $\mathbb{R}$ is given by the probability density function

$$f_\nu(\boldsymbol{x}) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi\nu}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{\boldsymbol{x}^2}{\nu}\right)^{-(\nu+1)/2}$$

where Γ is the gamma function. This includes Cauchy and normal distributions as special cases for $\nu = 1$ and $\nu \to \infty$ respectively.

A multivariate definition of the location-scale family (over $\mathbb{R}^n$ for a fixed n) can be given by appropriate parameters $(\mu, \Sigma) : 1 \to n \times \mathrm{Sym}_+(n)$, which gives us homs into n. The resulting category of kernels in the location-scale family will result in the same data as **Gauss**, but their morphisms will be interpreted as different distributions.

Definition 4.2.24. Let D be a distribution in the location-scale family closed under compound distributions. In the category $\mathcal{LS}_D$ of Euclidean spaces and conditional D distributions, objects are in the comonoid $(\mathbb{N}, \Delta)$. A morphism $(f, \Sigma, \mu) : n \to m$ consists of a linear map $f : \mathbb{R}^n \to \mathbb{R}^m$, a scale parameter $\Sigma : \mathrm{Sym}(1) \to \mathrm{Sym}(m)$ and a location parameter $\mu : 1 \to m$.

The mean and variance of a gaussian distribution coincide with the location and scale parameters of $\mathcal{LS}_\mathcal{N}$, so there is an equivalence $\mathcal{LS}_\mathcal{N} \cong$ **Gauss**. However, these parameters may encode different information; in example 4.2.22, the scale parameter encodes the bounds or support of the distribution. The aim of $\mathcal{LS}_D$ is thus to define a synthetic version of the category of gaussian preserving transformations $\mathbf{DF}_{\mathcal{N}_\mu}$ [144].

Proposition 4.2.25. *Let D be a distribution in the location-scale family. Pushforward of measures along affine functions are still in D.*

Proposition 4.2.26. *Let* $\mathbf{Gauss}_\bullet \to \mathbf{Euc}_{\mathrm{Aff}}$ *be the strong monoidal functor given by* $(f, \Sigma, \mu) \to (f, \mu)$ *(cf. [67]). This functor is an opfibration.*

Proof. Let $(f, a) : n \to m$ be an affine function that maps $x : \mathbb{R}^n$ to $(f, a)(x) := fx + a$ in $\mathbb{R}^m$, i.e. a morphism in $\mathbf{Euc}_{\mathrm{Aff}}$. Let $X : D\mathbb{R}^n$ be a covariance matrix over n. This characterises the equivalence class of all gaussian distributions over n quotiented by their mean. The cartesian lift of (f, a) is given by the Markov kernel $(f, 0, a) : n \to m$

in **Gauss** which corresponds to adding a Dirac distribution $Y = fX + \xi$ with $\mathbb{E}[\xi] = a$ and $\mathrm{Var}[\xi] = 0$ the zero variance matrix.

The indexed category is a functor

$$\mathcal{N} : \mathbf{Euc}_{\mathrm{Aff}} \to \mathbf{Pos} \hookrightarrow \mathbf{Cat} \tag{4.10}$$

(cf. example 2.3.5) of pointed gaussian distributions. $\mathcal{N}$ maps $n : \mathbf{Euc}_{\mathrm{Aff}}$ to the space $\mathcal{N}(n) := \mathrm{Sym}_+(n)$ of covariance matrices with Löwner order, and $(f, a) : n \to m$ to the (degenerately monotone) $\mathcal{N}(f, a) := \mathrm{const}_0 : \mathrm{Sym}_+(n) \to \mathrm{Sym}_+(m)$ that maps every covariance over $\mathbb{R}^n$ to the bottom element of $\mathrm{Sym}_+(m)$, the zero variance matrix. A morphism in $\mathbf{Gauss} = \int \mathcal{N}$ from $(n : \mathbf{Euc}_{\mathrm{Aff}}, \Theta : \mathcal{N}(n))$ to $(m : \mathbf{Euc}_{\mathrm{Aff}}, \Sigma : \mathcal{N}(m))$ consists of an affine function $(f, a) : n \to m$ such that $\mathcal{N}(f, a)(\Theta) \to \Sigma$ in $\mathcal{N}(m)$, which exists as $\mathcal{N}(f, a)(\Theta) = 0 \leq \Sigma$. This is equivalent to the gaussian kernel $(f, \Sigma, a) : n \to m$. □

4.3 Controlled dynamical systems

In the classical control literature [102], defining a dynamical system begins by fixing a single state space $X \cong \mathbb{R}^n$, over which the discrete dynamics are a set of $\mathbb{N}$-indexed maps called the *flow* or transition matrices $\{A(t)\}_{t \in \mathbb{N}}$ from the space X to itself such that $A(t + s) = A(t)A(s)$. This forms a semigroup homomorphism $\mathbb{N} \to \mathrm{End}(n)$, where the monoid identity is not commonly enforced. Controlled dynamical systems are represented via SSMs (example 2.5.11), and are said to be either open-loop and closed-loop. Open-loop systems refer to those whose control does not depend on the current state. For example, those with a desired output that do not get any observation of the real state, whose state space equations are given by

$$\boldsymbol{x}' = A\boldsymbol{x} + B\boldsymbol{u} \tag{4.11}$$

for linear maps $A : X \to X$, $B : U \to X$ and elements $x, x' \in X$ and $u \in U$. Closed-loop systems on the other hand consist of controls that depend on the systems state. Their

control laws are thus maps $K : X \to U$ from a state space to a control space, and their state space equations are given by

$$\begin{aligned} \boldsymbol{x}' &= A\boldsymbol{x} + B\boldsymbol{u} \\ \boldsymbol{u} &= K\boldsymbol{x} \end{aligned} \tag{4.12}$$

Our approach differs from this semigroup approach by treating discrete linear dynamical systems as linear maps $A : X \to Y$ between potentially different state spaces, and relying on the structure of the category **FVec** to cover the monoid formalisation as the case of endomorphisms over a fixed object X.

There are several formalisations of the concept of $\mathbb{N}$-indexed maps, including arrow categories (categories indexed by the interval category, which consists of two objects and one nontrivial morphism between them $I = \{0 \to 1\}$), categories indexed by a finite total order $N = \{0 \to 1 \to \cdots \to N\}$ or by the delooping of a monoid (definition A.1.2), and path categories of a quiver [165].

We choose an N-indexed category of processes to model linear time-variant systems, where not only the dynamics $A(t)$ depend on time, but the state spaces $X(t)$ themselves too. We generally refer to **processes** as morphisms and **systems** as indexed collections of (composable) processes.

Remark 4.3.1. In general, the action space at each time step U might depend on the state itself. Environment-based constraints are an example of this (see example 4.3.3). We say that the space of actions is a vector bundle fibred over the space of states, under appropriate smoothness of transition functions. However, capturing this bundle projection $p : U \to X$ as a morphism requires the ambient category to be **Top**. For the sake of concreteness, we focus on the trivial bundle $\pi_n : X \oplus U \to X$, which continues to be a morphism $\pi_n : n + u \to n$ in $\mathbf{FVec}_k$.

Definition 4.3.2 (Open-loop process)**.** An *open-loop process* in a category $\mathcal{C}$ is a span $X \leftarrow U \to Y$, with the left leg being a bundle projection and the right leg being the map $f : U \to Y$ which encode the dynamics. An identity open-loop process corresponds to the identity in $\mathbf{Span}(\mathcal{C})$: $X \overset{\text{id}}{\leftarrow} X \overset{\text{id}}{\to} X$.

Example 4.3.3. Consider the gridworld example from section 3.3.2, where an agent moves in the four cardinal directions. The constraint imposed by walls preventing to advance at certain points in the state space is captured by an action space bundled over the state space in a non-trivial way; the set of actions in the corner states have two elements, three elements in the wall segments and four in any middle state, defining an open-loop process $X \overset{\pi_X}{\leftarrow} \sum_{x:X} A(x) \to Y$ whose left leg is the projection of the state from the dependent sum of state-action pairs.

Example 4.3.4 (Linear open-loop process)**.** A linear open-loop process is an open-loop process $n \leftarrow n + u \to m$ in $\mathbf{LinSpan}_{\mathbb{R}}$. The control does not depend on the values of the state, i.e. u is a constant type family over n, the apex is the (bi)product $n + u$, and the dynamics are a pair (A, B) of linear maps $A : \mathbb{R}^n \to \mathbb{R}^m$ and $B : \mathbb{R}^u \to \mathbb{R}^m$ as in (4.11).

Definition 4.3.5. A (finite) *discrete linear time-variant system* (LTV) is an N-indexing of $\mathbf{Para}_+(\mathbf{FVec})$. Concretely, this is a sequence of composable controllable processes $\{(U_i, A_i, B_i) : X_i \to X_{i+1}\}_{i=0}^N$ with state spaces X_i, control spaces U_i and dynamics given by $x_{i+1} = A_i x_i + B_i u_i$.

An LTV is controllable when the controllability Gramian is non-singular.

Definition 4.3.6. The *controllability Gramian* of an LTV $\{(U_i, A_i, B_i)\}_{i=0}^N$ is the $N \times N$ square matrix whose (i, j) entry for $0 \leq i < j \leq N$ is given by

$$\mathcal{C}(i, j) = \sum_{k=i}^{j} \phi(j, k) B_k B_k^\top \phi^\top(j, k)$$

where $\phi(j, k) = A_k \circ A_{k-1} \circ \cdots A_j$.

In time-invariant systems, where X_i, U_i, A_i, B_i are constant for all i, $\mathcal{C}(i, j)$ is inductively defined both in j (starting at i, forwards) and i (starting at j, backwards). However, in LTVs the time dependence on A_i and B_i makes only the second property hold. For the last time step $\mathcal{C}(j, j) = B_j B_j^\top$, and for $i < j$,

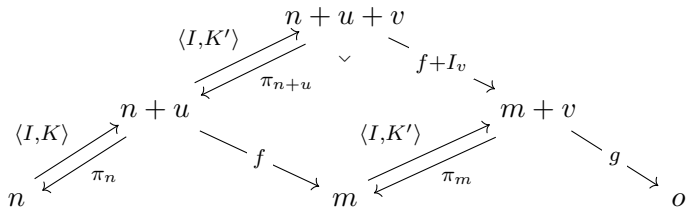
$$\mathcal{C}(i, j) = \mathcal{C}(i + 1, j) + \phi(j, i) B_i B_i^\top \phi^\top(j, i)$$

Definition 4.3.7 (Closed-loop process)**.** A *closed-loop process* or *state-space model with feedback* is an open-loop process $n \leftarrow u \rightarrow m$ with a choice of section or right-inverse $K : n \to u$ of the left leg of the span. This section is called the *control law*.

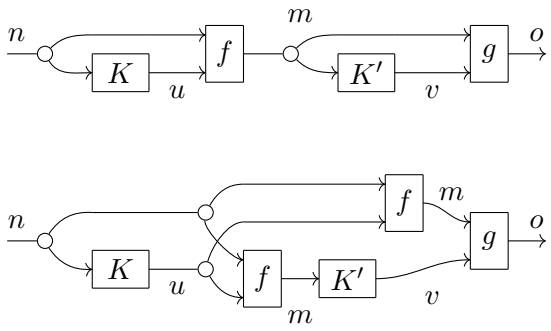
Proposition 4.3.8. *There's a forgetful functor from linear closed-loop systems to* **FVec** *that maps closed-loop processes* $n \underset{\pi_n}{\overset{\langle I,K\rangle}{\rightleftarrows}} n+u \xrightarrow{f} m$ *to linear maps*

$$f \circ \langle I, K\rangle : n \to m \tag{4.13}$$

Proof. The identity linear closed-loop process $X \underset{\pi_n}{\overset{\langle I,!\rangle}{\rightleftarrows}} n+0 \xrightarrow{I} n$ is sent to $I \circ \langle I, !\rangle = \mathrm{id}_n$. Two linear closed-loop processes compose via pullback as



Its image under the functor is the linear map $g \circ (f + I_v) \circ \langle I, K'f\rangle \circ \langle I, K\rangle : n \to o$ is the composition of $f \circ \langle I, K\rangle : n \to m$ with $g \circ \langle I, K'\rangle : m \to o$, as shown in the following string diagrams in $(\mathbf{FVec}, +, 0)$ (see section 2.4):



□

4.4 Linear quadratic regulators

A discrete linear quadratic regulator (LQR) is a dynamical system where the state and control at each step k have an associated *immediate cost* defined by a quadratic

function $\ell_k : X_k \oplus U_k \to \mathbb{R}$, given by a matrix S_k and two positive definite matrices Q_k and R_k:

$$\ell_k(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{x}_k^\top Q_k \boldsymbol{x}_k + \boldsymbol{u}_k^\top R_k \boldsymbol{u}_k + \boldsymbol{x}_k^\top S_k \boldsymbol{u}_k + \boldsymbol{u}_k^\top S_k^\top \boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{x}_k^\top & \boldsymbol{u}_k^\top \end{bmatrix} \begin{bmatrix} Q_k & S_k \\ S_k^\top & R_k \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{bmatrix}$$

The typical optimization problem to be solved is the search of a control law K_k that transforms the open-loop system to a closed-loop system that minimizes a cumulative cost quantity $J \in \mathrm{Sym}_+(X)$ (cf. (4.6))

$$J(\boldsymbol{x}_0) = \left\{ \sum_{k=0}^{N} \ell(\boldsymbol{x}_k, \boldsymbol{u}_k) \;\middle|\; \begin{array}{c} \boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k \\ \boldsymbol{u}_{k+1} = K_k \boldsymbol{x}_k \end{array} \right\} \tag{4.14}$$

up to some step $N : \mathbb{N}$. It is the archetypal problem studied in the field of optimal control theory, and can be found in any introductory textbook [66, Ch.9][154, Ch.3][4].

The search for a minimum cost J is a global optimization problem, meaning that it depends on the whole trajectory $X^{\oplus N} \oplus U^{\oplus N}$ and the actions taken at each step $K_k : X_k \to U_{k+1}$. However, the quadratic nature of the cost allows to decompose it via various methods to the sum of independent local problems. One of the methods is the Riccati equation, which solves for the optimal P_k (which determines $J_k(\boldsymbol{x}) = \boldsymbol{x}^\top P_k \boldsymbol{x}$) via the expression[4]:

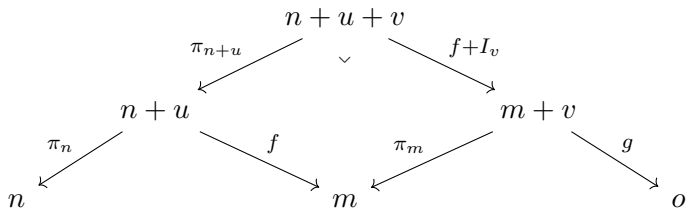
$$P_k = Q + A^\top P_{k+1} A - (S^\top + A^\top P_{k+1} B)(R + B^\top P_{k+1} B)^{-1}(S + B^\top P_{k+1} A) \tag{4.15}$$

Before we characterise this equation in section 4.4.1, we first need a language to talk about cost functions on dynamical systems, and their composition, which is going to be the category $\mathrm{Sym}_+^{\mathsf{op}}$ **Span**.

Example 4.4.1. Consider two steps of an LQR problem, defined by dynamics $f : n + u \to m$, $g : m + v \to o$ and associated quadratic costs $P_{n+u} : \mathrm{Sym}_{\bullet+}(n + u)$ and $P_{m+v} : \mathrm{Sym}_{\bullet+}(m + v)$. These define two composable morphisms $n \xleftarrow{\pi_n} n + u \xrightarrow{f} m$ with structure element $P_{n+u} : 1 \to \mathrm{Sym}_+(n+u)$ and $m \xleftarrow{\pi_m} m+v \xrightarrow{g} o$ with structure element

[4] The derivation of this equation can be found in section A.2.1

$P_{m+v} : 1 \to \mathrm{Sym}_+(m+v)$ in $\mathrm{Sym}_+^{\mathsf{op}}$ **Span**. Their composition is given by the pullback



Because we are working with spans whose left legs are projections, the expression of the apex, calculated by the graph of f (definition A.1.3), simplifies to $\mathrm{gr}(f)+v \cong n+u+v$. The structure element $P_{\mathrm{gr}(f)+v} : 1 \to \mathrm{Sym}_+(\mathrm{gr}(f)+v)$ is given by

$$\begin{aligned} P_{n+u+v} : 1 &\xrightarrow{\lambda^{-1}} 1+1 \\ &\xrightarrow{P_{n+u}+P_{m+v}} \mathrm{Sym}_+(n+u)+\mathrm{Sym}_+(m+v) \\ &\xrightarrow{\nabla_{n+u,m+v}} \mathrm{Sym}_+(n+u+m+v) \\ &\xrightarrow{\mathrm{Sym}_+[\pi_{n+u},f+1_v]} \mathrm{Sym}_+(n+u+v) \end{aligned}$$

which encodes the addition of both cost functions under the constraint of dynamics. In a more traditional notation, this reads: For all vectors $\boldsymbol{x} \in X \cong \mathbb{R}^n, \boldsymbol{u} \in U \cong \mathbb{R}^u, \boldsymbol{v} \in V \cong \mathbb{R}^v$,

$$P_{n+u+v}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v}) = P_{n+u}(\boldsymbol{x},\boldsymbol{u}) + P_{m+v}(f(\boldsymbol{x},\boldsymbol{u}),\boldsymbol{v})$$

and point-free:

$$P_{n+u+v} = (\pi_{n+u})^\top P_{n+u}\pi_{n+u} + (f+1_v)^\top P_{m+v}(f+1_v) \tag{4.16}$$

Following section 4.2.1, this composition of P_{n+u} and P_{m+v} specifies a cost function on the space $\mathbb{R}^n \oplus \mathbb{R}^u \oplus \mathbb{R}^v$, which consists of the state and control spaces $\mathbb{R}^n \oplus \mathbb{R}^u$, the control space $\mathbb{R}^v$ but only the subspace of $\mathbb{R}^m$ in the image of the dynamics f. An explicit matrix representation for P_{n+u+v} will be shown in theorem 4.4.3.

On the other hand, even though the weighted para construction composes quadratic costs (as 'weights') via the appropriate laxator, it cannot express quadratic costs P_{n+u} over $n+u$, only $P_n : F \circ \mathrm{Sym}_{\bullet+}(n)$ as a set (after the forgetful $F : \mathbf{SMod}_{\mathbb{R}+} \to \mathbf{Set}$).

4.4.1 Compositional Riccati equations

At the end of section 4.2, theorem 4.2.16 showed how $\mathbf{El}(\mathrm{Sym}^{\mathsf{op}}_{\bullet+})$ picks optimal estimations of non-cumulative costs in $\int \mathrm{Sym}_{\bullet+}$ for uncontrolled linear dynamics. Example 4.2.17 gave a category where accumulated costs can be compared. The formalisation of open-loop systems as $\mathbf{LinSpan}_{\mathbb{R}}$ morphisms and costs as their decorations gives us the necessary machinery to talk about the optimal estimation of accumulated costs.

Lemma 4.4.2. *The Riccati difference equation (RDE) (4.15) defines a smooth (non-linear) operator* $\mathrm{Sym}_+(m) \to \mathrm{Sym}_+(n)$ *when* R *is positive definite.*

Proof. The set $\mathrm{Sym}_\bullet(n)$ is smooth manifold that is globally diffeomorphic to $\mathbb{R}^{n(n+1)/2}$. The space $\mathrm{Sym}_{\bullet+}(n)$ is an open subset of $\mathrm{Sym}_\bullet(n)$, thus also a smooth manifold. We have to show that the following expression is smooth in $P_m \in \mathrm{Sym}_{\bullet+}(m)$.

$$Q + A^\top P_m A - (S^\top + A^\top P_m B)(R + B^\top P_m B)^{-1}(S + B^\top P_m A) \tag{4.17}$$

$Q + A^\top P_m A$ is affine in P_m. P_m is positive semi-definite so $(R + B^\top P_m B)$ is strictly invertible when R is positive definite. Sums and products of smooth terms are smooth. □

To better understand the matrix expression of the Riccati equation, it is useful to separate its derivation into two distinct steps:

- Constraint pullback: Given immediate state-action costs P_{n+u}, future state costs P_m and dynamics $f : \mathbb{R}^n \oplus \mathbb{R}^u \to \mathbb{R}^m$, we can give explicit expressions for their block matrix decompositions. By the universal properties of the biproduct in $(\mathbf{FVec}, +, 0)$, let $\pi_u : n+u \to u$, $\pi_n : n+u \to n$, $\iota_u : u \to n+u$ and $\iota_n : n \to n+u$ be the projection and injection maps for control and state spaces.

  The block matrices $P_{n+u} = \begin{pmatrix} Q & S \\ S^\top & R \end{pmatrix}$ and $f = \begin{pmatrix} A & B \end{pmatrix}$ are given by

$$\begin{aligned} Q &:= \pi_n \circ P_{n+u} \circ \iota_n & A &:= f \circ \iota_n \\ S &:= \pi_n \circ P_{n+u} \circ \iota_u & B &:= f \circ \iota_u \\ R &:= \pi_u \circ P_{n+u} \circ \iota_u \end{aligned} \tag{4.18}$$

which satisfy $f(\boldsymbol{x},\boldsymbol{u}) = A\boldsymbol{x} + B\boldsymbol{u}$ and $(\boldsymbol{x},\boldsymbol{u})^\top P_{n+u}(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{x}^\top Q\boldsymbol{x} + \boldsymbol{x}^\top S\boldsymbol{u} + \boldsymbol{u}^\top S^\top \boldsymbol{x} + \boldsymbol{u}^\top R\boldsymbol{u}$. Note that by P_{n+u} being symmetric, $(\pi_n \circ P_{n+u} \circ \iota_u)^\top = (\iota_u)^\top \circ P_{n+u}^\top \circ (\pi_n)^\top = \pi_u \circ P_{n+u} \circ \iota_n$, thus $S^\top$ is well-defined.

Following section 4.2.1, we compute the cumulative current costs (which we will denote P^*_{n+u} to differentiate it from the immediate costs P_{n+u}) as:

$$P^*_{n+u} = P_{n+u} + f^\top P_m f = \begin{bmatrix} Q + A^\top P_m A & S + A^\top P_m B \\ S^\top + B^\top P_m A & R + B^\top P_m B \end{bmatrix} \tag{4.19}$$

- Schur complement: The optimal current state cost is given by the Schur complement (definition A.2.2) of the cost associated to the control space $\mathbb{R}^u$ in P^*_{n+u}

$$P^*_n = P^*_{n+u}/(R + B^\top P_m B) \tag{4.20}$$

The theorem presented next shows that the Riccati equation picks the optimal cost of the control problem, just like $\mathbf{El}(\mathrm{Sym}^{\mathsf{op}}_{\bullet+})$ picked the optimal cost for a non-controlled system in theorem 4.2.16. However, because the Schur complement step is not a linear but smooth expression of P (lemma 4.4.2), our target category is not $\mathbf{FVec}_{\mathbb{R}}$. For now, we choose to track cost functions (elements of the object $\mathrm{Sym}_+(n)$) as quadratic forms with vector space structure, so we define the codomain category to consist of (pointed) euclidean spaces and smooth maps. If we insisted to also retain their positive semi-definiteness, the category $\mathbf{Smooth}_\bullet$ would not be enough, as an object $n(n+1)/2$ only generates freely a space $\mathbb{R}^{n(n+1)/2} \cong \mathrm{Sym}(n)$ and the subspace $\mathrm{Sym}_+(n)$ can only be characterised as an object of $\mathbf{Mfd}_\infty$.

Theorem 4.4.3. *The Riccati difference equation defines a functor* $\mathrm{RD} : (\mathrm{Sym}^{\mathsf{op}}_+ \mathbf{Span})^{\mathsf{op}} \to \mathbf{Smooth}_\bullet$, *that sends objects* n *to* optimal *cost functions*[5] $\mathrm{RD}(n) = (n(n+1)/2, P^*_n)$ *and decorated spans* $(\pi_n, f, P_{n+u}) : n \to m$ *to the smooth operator* $\mathrm{RD}(\pi_n, f, P_{n+u})$ *given by* (4.17).

[5]Recall the dimension of the space of positive semi-definite matrices $\mathrm{Sym}_+(n)$ is $n(n+1)/2$. This natural number is the object in **Smooth**.

Proof. Let $(\pi_n, f, P_{n+u}) : n \leftarrow n+u \rightarrow m$ be a morphism in $\mathrm{Sym}_+^{\mathsf{op}}$ **Span**. $\mathrm{RD}(\pi_n, f, P_{n+u})$ is given by (4.17) where Q, R, S, A, B are calculated via the block matrix decompositions (4.18). By lemma 4.4.2, $\mathrm{RD}(\pi_n, f, P_{n+u})$ is a morphism in **Smooth**, and by proposition A.2.7 it is a morphism in **Smooth**$_\bullet$ that given a point in $\mathrm{Sym}_+(m)$, picks an optimal point in $\mathrm{Sym}_+(n)$.

The functoriality is proven by two preservation properties:

- Preservation of identity: Let $(\mathrm{id}_n, \mathrm{id}_n, 0_n)$ be the identity morphism on n in $\mathrm{Sym}_+^{\mathsf{op}}$ **Span**. The control space is the zero-dimensional space $U = \mathbb{R}^0$, and the zero cost function 0_n is defined over $n + 0 = n$.

$$\mathrm{RD}(\mathrm{id}_n, \mathrm{id}_n, 0_n)(P_n) = 0_n + \mathrm{id}_n^\top P_n \,\mathrm{id}_n = P_n \tag{4.21}$$

- Compositionality: Let $(\pi_n, f, P_{n+u}) : n \rightarrow m$ and $(\pi_m, g, P_{m+v}) : m \rightarrow o$, and denote their composition as $(\pi_n, g \circ (f + 1_v), P_{n+u+v}) : n \rightarrow o$. We need to show that the diagram (4.22)—following notation 2.6.6 composes to (4.23). In this diagram, the pullback symbol denotes the constraint pullback step (4.19) and Schur denotes the Schur complement step (4.20).

$$\begin{array}{ll}
\textbf{Smooth}_\bullet & P_n^* \xleftarrow{\text{Schur}} P_{n+u}^* \longleftarrow P_m^* \xleftarrow{\text{Schur}} P_{m+v}^* \longleftarrow P_o \\
 & \quad\uparrow \lrcorner \qquad\qquad\quad \uparrow \lrcorner \\
\mathrm{Sym}_+^{\mathsf{op}}\ \textbf{Span} & \quad P_{n+u} \qquad\qquad\quad P_{m+v} \\
\textbf{FVec}_{\mathbb{R}} & n \xleftarrow{\pi_n} n+u \xrightarrow{f} m \xleftarrow{\pi_m} m+v \xrightarrow{g} o
\end{array} \tag{4.22}$$

$$\begin{array}{ll}
\textbf{Smooth}_\bullet & P_n^* \xleftarrow{\text{Schur}} P_{n+u+v}^* \longleftarrow P_o \\
 & \quad\uparrow \lrcorner \\
\mathrm{Sym}_+^{\mathsf{op}}\ \textbf{Span} & \quad P_{n+u+v} \\
\textbf{FVec}_{\mathbb{R}} & n \xleftarrow{\pi_n} n+u+v \xrightarrow{g\circ(f+1_v)} o
\end{array} \tag{4.23}$$

Equationally, this consists of showing the equality of matrix expressions

$$\mathrm{RD}(\pi_n, f, P_{n+u}) \circ \mathrm{RD}(\pi_m, g, P_{m+v})(P_o) = \mathrm{RD}(\pi_n, g \circ (f + 1_v), P_{n+u+v})(P_o) \quad (4.24)$$

This is proved pointwise for all $P_o : \mathrm{Sym}^*_+(o)$ next. Let $P_{n+u} = \left(\begin{smallmatrix} Q & S \\ S^\top & R \end{smallmatrix}\right)$ and $P_{m+v} = \left(\begin{smallmatrix} Q' & S' \\ S'^\top & R' \end{smallmatrix}\right)$. We expand the expressions of each side of the equation and show their equality.

LHS: Given $P_o : \mathrm{Sym}_{\bullet+}(o)$, the computation of P^*_n via the left hand side of (4.24) following (4.22) goes by substituting $P^*_m = \mathrm{RD}(\pi_m, g, P_{m_v})(P_o)$ into the argument of $\mathrm{RD}(\pi_n, f, P_{n+u})$:

$$\begin{aligned} P^*_m &= Q' + A'^\top P^*_o A' - (S'^\top + A'^\top P^*_o B')(R' + B'^\top P^*_o B')^{-1}(S' + B'^\top P^*_o A') \\ P^*_n &= Q + A^\top P^*_m A - (S^\top + A^\top P^*_m B)(R + B^\top P^*_m B)^1(S + B^\top P^*_m A) \end{aligned} \quad (4.25)$$

$$\begin{aligned} &= Q + A^\top Q' A + A^\top A'^\top P_o A' A \\ &- A^\top (S'^\top + A'^\top P_o B')(R' + B'^\top P_o B')^{-1}(S' + B'^\top P_o A')A \\ &- (S^\top + A^\top Q' B + A^\top A'^\top P_o A' B - A^\top (S'^\top + A'^\top P_o B')(R' + B'^\top P_o B')^{-1}(S' + B'^\top P_o A')B) \\ &\cdot (R + B^\top Q' B + B^\top A'^\top P_o A' B - B^\top (S'^\top + A'^\top P^*_o B')(R' + B'^\top P^*_o B')^{-1}(S' + B'^\top P^*_o A')B)^{-1} \\ &\cdot (S + B^\top Q' A + B^\top A'^\top P_o A' A - B^\top (S'^\top + A'^\top P^*_o B')(R' + B'^\top P^*_o B')^{-1}(S' + B'^\top P^*_o A')A) \end{aligned}$$

RHS: The matrix representation of P_{n+u+v} composed as in (4.16) under a basis of $\mathbb{R}^n \oplus \mathbb{R}^u \oplus \mathbb{R}^v$ is given by

$$\begin{aligned} P_{n+u+v} &= (\pi_{n+u})^\top P_{n+u} \pi_{n+u} + (f + 1_v)^\top P_{m+v}(f + 1_v) \\ &= \begin{bmatrix} I & 0 \\ 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \end{bmatrix} + \begin{bmatrix} A^\top & 0 \\ B^\top & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} Q' & S' \\ S'^\top & R' \end{bmatrix} \begin{bmatrix} A & B & 0 \\ 0 & 0 & I \end{bmatrix} \\ &= \begin{bmatrix} Q + A^\top Q' A & S + A^\top Q' B & A^\top S' \\ S^\top + B^\top Q' A & R + B^\top Q' B & B^\top S' \\ S'^\top A & S'^\top B & R' \end{bmatrix} \end{aligned}$$

Now we calculate $\mathrm{RD}(\pi_n, g \circ (f + 1_v), P_{n+u+v})(P_o)$ following (4.23) in two steps: Firstly, the cumulative costs over $n + u + v$ via constraint pullback:

$$
\begin{aligned}
P^*_{n+u+v} &= P_{n+u+v} + (g \circ (f + 1_v))^\top P_o (g \circ (f + 1_v)) \\
&= \begin{bmatrix} Q + A^\top Q' A & S + A^\top Q' B & A^\top S' \\ S^\top + B^\top Q' A & R + B^\top Q' B & B^\top S' \\ S'^\top A & S'^\top B & R' \end{bmatrix} + \begin{bmatrix} A^\top A'^\top \\ B^\top A'^\top \\ B'^\top \end{bmatrix} P_o \begin{bmatrix} A'A & A'B & B' \end{bmatrix} \\
&= \begin{bmatrix} Q + A^\top Q' A + A^\top A'^\top P_o A' A & S + A^\top Q' B + A^\top A'^\top P_o A' B & A^\top S' + A^\top A'^\top P_o B' \\ S^\top + B^\top Q' A + B^\top A'^\top P_o A' A & R + B^\top Q' B B^\top A'^\top P_o A' B & B^\top S' + A^\top A'^\top P_o B' \\ S'^\top A + B'^\top P_o A' A & S'^\top B + B'^\top P_o A' B & R' + B'^\top P_o B' \end{bmatrix}
\end{aligned}
$$

Secondly, P^*_n is given by the Schur complement of the lower left 2×2 block. By lemma A.2.4, the equality with (4.25) is immediate:

$$
\begin{aligned}
P^*_n = \; & P^*_{n+u+v} / \begin{pmatrix} R + B^\top Q' B B^\top A'^\top P_o A' B & B^\top S' + A^\top A'^\top P_o B' \\ S'^\top B + B'^\top P_o A' B & R' + B'^\top P_o B' \end{pmatrix} \\
= \; & Q + A^\top Q' A + A^\top A'^\top P_o A' A \\
& - (S^\top + A^\top Q' B + A^\top A'^\top P_o A' B - (A^\top S'^\top + A^\top A'^\top P_o B')(R' + B'^\top P_o B')^{-1}(S' B + B'^\top P_o A' B)) \\
& \cdot (R + B^\top Q' B + B^\top A'^\top P_o A' B - (B^\top S'^\top + B^\top A'^\top P^*_o B')(R' + B'^\top P^*_o B')^{-1}(S' B + B'^\top P^*_o A' B))^{-1} \\
& \cdot (S + B^\top Q' A + B^\top A'^\top P_o A' A - (B^\top S'^\top + B^\top A'^\top P^*_o B')(R' + B'^\top P^*_o B')^{-1}(S' A + B'^\top P^*_o A' A)) \\
& - (A^\top S'^\top + A^\top A'^\top P_o B')(R' + B'^\top P_o B')^{-1}(S' A + B'^\top P_o A' A)
\end{aligned}
$$

□

4.4.2 The extended Kalman filter

A similar functor can be defined for optimal estimation of observed processes, in particular, the Kalman filter (KF) for linear systems. Let $\mathbb{R}^n$ and $\mathbb{R}^u$ be fixed state and observation spaces. From a linear observed process $f : n \to n + u$ given by matrices

$A : n \to n$ and $C : n \to u$, KF computes the optimal correction from observations in $\mathbb{R}^u$ to transform prior estimations of the state in $\mathbb{R}^n$ to posterior estimations such that the estimation covariance is minimal. The estimation covariance $\Sigma_{k+1|k+1}$ is a symmetric positive semi-definite matrix that characterises the Gaussian distribution of the error between the estimation and the true state. We subscript it with $k+1|k+1$ to clarify that it is the error for step $k+1$ given estimations up to step $k+1$. A derivation of the Kalman filter algorithm and the optimal estimation covariance $\Sigma^*_{k+1|k+1}$ is given in the appendix (section A.2.2).

The extended Kalman filter (EKF) [94, 119] adapts this algorithm to non-linear models $\boldsymbol{x}' = f(k, \boldsymbol{x})$, $\boldsymbol{y} = g(k, \boldsymbol{x})$ by a time-variant linearisation

$$\begin{aligned} \boldsymbol{x}_{k+1} &= A_k \boldsymbol{x}_k + \boldsymbol{w}_k \\ \boldsymbol{y}_k &= C_k \boldsymbol{x}_k + \boldsymbol{v}_k \end{aligned} \tag{4.26}$$

where the linear maps are given by $A_k = \partial_{\boldsymbol{x}} f(k, \boldsymbol{x})|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k}$ and $C_k = \partial_{\boldsymbol{x}} g(k, \boldsymbol{x})|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k}$ and $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are zero-mean gaussian noise with covariances $\mathbb{E}[\boldsymbol{w}_k \boldsymbol{w}_k^\top] = W$ and $\mathbb{E}[\boldsymbol{v}_k \boldsymbol{v}_k^\top] = V$. A time-variant observed process (recall section 2.5.2) will be a linear cospan $n \xrightarrow{f} m+u \xleftarrow{\iota_m} m$, whose left leg f consists of the dynamics map $A : n \to m$ and the observation map $C : n \to u$. The right leg $\iota_m : m \to m+u$ is simply the inclusion of the next state into the apex. This linear cospan holds the deterministic fragment of the system (4.26), while we choose to hold the noise covariance data as a Sym_+-decoration $\Sigma_{m+u} := \left(\begin{smallmatrix} W & 0 \\ 0 & V \end{smallmatrix}\right)$ on the apex $m+u$ of the morphism. The derivation of the optimal gain matrix (the *Kalman* gain) resulting in the optimal (that is, minimal) variance $\Sigma^*_{k+1|k+1}$ is given by solving the same Riccati equation as LQRs. The relation between Kalman filters and LQRs later will be discussed in section 4.4.4. We give next the definition of the Riccati matrix difference functor for estimation, which we denote with the same label RD.

Definition 4.4.4. Let $\mathrm{RD} : \mathrm{Sym}_+ \mathbf{Cospan} \to \mathbf{Smooth}_\bullet$ be the covariant functor that sends objects n to estimation covariances $\mathrm{RD}(n) = (n(n+1)/2, \Sigma^*_n)$ and decorated cospans $(f, \iota_m, \Sigma_{m+u}) : n \to m$ with $f = \left(\begin{smallmatrix} A \\ C \end{smallmatrix}\right)$ and $\Sigma_{m+u} = \left(\begin{smallmatrix} W & 0 \\ 0 & V \end{smallmatrix}\right)$ to the smooth

operator $\mathrm{RD}(f, \iota_m, \Sigma_{m+u})$ given by

$$\Sigma_n \mapsto A\Sigma_n A^\top + W - (A\Sigma_n A^\top + W)C^\top (C(A\Sigma_n A^\top + W)C^\top + V)^{-1} C(A\Sigma_n A^\top + W) \tag{4.27}$$

Theorem 4.4.5. *The functor* RD *is well-defined and computes the extended Kalman filter, that is, the optimal estimation of states of a linearised observed dynamical system.*

Proof. Let Σ_n be an initial covariance matrix, and $\Sigma_{m+u} = \left(\begin{smallmatrix} W & 0 \\ 0 & V \end{smallmatrix}\right)$ the covariance associated to the zero-mean gaussian noise in (4.26). Let $f = \left(\begin{smallmatrix} A \\ C \end{smallmatrix}\right)$ denote the deterministic component of the dynamics.

$$\begin{array}{ll} \mathbf{Smooth}_\bullet & \Sigma_n \longmapsto \Sigma^*_{m+u} \xrightarrow{\text{Schur}} \Sigma^*_m \\ & \quad\llcorner \quad\uparrow \\ \mathrm{Sym}_+ \mathbf{Cospan} & \qquad\quad \Sigma_{m+u} \\ & \\ \mathbf{FVec}_{\mathbb{R}} & n \xrightarrow{f} m+u \xleftarrow{\iota_m} m \end{array} \tag{4.28}$$

The cumulative covariance Σ^*_{m+u} is given by:

$$\Sigma^*_{m+u} = \begin{bmatrix} W + A\Sigma_n A^\top & (A\Sigma_n A^\top + W)C^\top \\ C(A\Sigma_n A^\top + W) & V + C(A\Sigma_n A^\top + W)C^\top \end{bmatrix}$$

The Schur complement step computes:

$$\begin{aligned} \Sigma^*_m &= \Sigma^*_{m+u}/(V + C\Sigma_n C^\top) \\ &= W + A\Sigma_n A^\top - (A\Sigma_n A^\top + W)C^\top (V + C(A\Sigma_n A^\top + W)C^\top)^{-1} C(A\Sigma_n A^\top + W) \end{aligned}$$

which results in (4.27). Compositionality and preservation of identity is proved analogously to theorem 4.4.3. □

4.4.3 The LTI monoid

Having shown the compositionality of Riccati differences for time-variant systems, their time-invariant (LTI) case falls out of the restriction of RD to the monoid of decorated

spans over a fixed state space. The fixpoint of the (endomorphic) Riccati difference equation is known as the *discrete algebraic Riccati equation* (definition A.2.9).

Definition 4.4.6 (Riccati evolution map [51])**.** Let $n : \mathbf{FVec}_{\mathbb{R}}$ be a fixed vector space $\mathbb{R}^n$, $A : n \to n$ a square matrix representing a time-invariant system, and $G, H \in \mathrm{Sym}_{\bullet +}(n)$ two immediate cost matrices. The forward and backward evolution maps are:

$$\begin{aligned} \Phi : \mathrm{Sym}_+(n) &\to \mathrm{Sym}_+(n) & \hat{\Phi} : \mathrm{Sym}_+(n) &\to \mathrm{Sym}_+(n) \\ P &\mapsto A(I + PH)^{-1}PA^\top + G & P &\mapsto A^\top(I + PG)^{-1}PA + H \end{aligned} \tag{4.29}$$

Lemma 4.4.7. *The Riccati evolution map* Φ *defines the* forward Riccati evolution semigroup $\{\Phi_k\} \subseteq \mathrm{End}(\mathrm{Sym}_+(n))$ *whose elements consist of* k *times iterated composition of* Φ*. The semigroup operation is operator composition.*

Although seldom mentioned in semigroup theory, $\{\Phi_k\}$ is a (noncommutative) monoid with the identity operator as the neutral element $\Phi_0 := \mathrm{id}_{\mathrm{Sym}_+(n)}$, so we will talk about the *forward Riccati evolution monoid*. The operator $\mathrm{id}_{\mathrm{Sym}_+(n)}$ should not be confused with the identity matrix I in $\mathrm{Sym}_{\bullet +}(n)$. Similarly, $\hat{\Phi}$ defines the *backward Riccati evolution monoid*. The following corollaries to theorem 4.4.3 (resp. theorem 4.4.5) show how we can generalize Riccati monoids to 'Riccati categories'.

Corollary 4.4.8. *The backward Riccati evolution map* (4.29) *is the image of endomorphisms in* $\mathrm{Sym}_+^{\mathsf{op}}$ **Span** *with zero state-control costs* $S = 0$ *under* RD*.*

Proof. Let $(\pi_n, f, P_{n+u}) : n \leftarrow n + u \to n$ be an endomorphism in $\mathrm{Sym}_+^{\mathsf{op}}$ **Span**. Given the same decomposition of f and P_{n+u} into (A, B, Q, S, R) as (4.18), we have $S = 0$ by assumption. Then

$$\begin{aligned} &\mathrm{RD}(\pi_n, f, P_{n+u})(P_m) \\ &= Q + A^\top P_m A - (S^\top + A^\top P_m B)(R + B^\top P_m B)^{-1}(S + B^\top P_m A) \\ &= A^\top P_m (I + G P_m)^{-1} A + Q \qquad (\text{with } G = BR^{-1}B^\top \text{ by proposition A.2.10}) \\ &= A^\top (I + P_m G)^{-1} P_m A + Q \\ &= \hat{\Phi}(P_m) \qquad (\text{with } H = Q) \end{aligned}$$

The second to last equality is given by transposing the first term, and noting that for symmetric positive semi-definite P_m and G we have the equality $(I + P_m G)^{-1} = (I + G P_m)^{-1}$. □

Corollary 4.4.9. *The Kalman filter is given by the EKF for linear time-invariant systems. This corresponds to images of endomorphisms in* Sym_+ **Cospan** *under* RD.

The inverted pendulum example introduced in section 3.3.3 was modelled in terms of optics over smooth maps in example 3.5.2. Given a time discretization, the value improvement step for the propagator $\Phi : X \to X$ (3.15) consisted of the composition of two morphisms in $\mathbf{Optic}(\mathbf{Mfd}_\infty)$ with an action $u = Kx$ given by a *fixed* policy or control law $K : X \to U$. The improvement of the policy, or the computation of an optimal one, was handled extraneously by updating the value function. We now revisit this example in the framework of Sym_+-decorated spans, and illustrate how RD provides a functorial characterization of the Bellman equation in the context of LQRs.

Moreover, the general theory of non-endomorphic decorated spans allows computing optimal controls for linearisations out of equilibrium, and problems whose state or control spaces change over time.

Example 4.4.10 (Inverted pendulum out of equilibrium)**.** Let $M = \mathbb{R} \times \mathbb{R} \times \mathbb{S}^1 \times \mathbb{R}$, and $f : M \times \mathbb{R} \to TM$. Given a trajectory $((\boldsymbol{x}_0, u_0), (\boldsymbol{x}_1, u_1), \dots)$ of state-control pairs, the linearisation of $f(\boldsymbol{x}_k, u_k)$ at these different points by Taylor (3.16) is given by matrices $A_{\boldsymbol{x}_k, u_k}$, $B_{\boldsymbol{x}_k, u_k}$ that change over the course of the trajectory. These are only constant when the trajectory is near an equilibrium, so in general they constitute a discrete LTV (definition 4.3.5) $\{(U, A_k, B_k) : X_k \to X_{k+1}\}$ in $\mathbf{Para}_+(\mathbf{FVec})$ with constant control space U, or equivalently a sequence of composable morphisms in $\mathbf{LinSpan}_{\mathbb{R}}$ following proposition 2.5.6. Here we qualify the cost function $\tilde{\ell}_k : X_k \times U \to \mathbb{R}$ to be quadratic. Its associated Hessian, which we may call $P_{n_k+u_k} = \left(\begin{smallmatrix} Q & S \\ S^\top & R \end{smallmatrix}\right)$ for $n_k = \dim X_k$ and $u_k = \dim U$, decorates the sequence and defines morphisms $(\pi_{n_k}, f, P_{n_k+u_k}) : X_k \to X_{k+1}$ in $\mathrm{Sym}_+^{\mathsf{op}}$ **Span**.

By theorem 4.4.3, the images $\mathrm{RD}(\pi_{n_k}, f, P_{n_k+u_k})$ define the backward maps of optimal cost functions $P^*_{n_k}$, from which the optimal controls u^*_k at each time step k are

given by the linear control law $K_k^* : X_k \to U$ defined according to (1) as:

$$K_k^* := -(R_k + B_k^\top P_n^* B_k)^{-1}(S_k + B_k^\top P_n^* A_k)$$

The forgetful functor to **FVec** (proposition 4.3.8) defines the *optimal trajectory* of the inverted pendulum that minimizes the cumulative cost as the closed-loop system.

4.4.4 The control-estimation duality

Kalman's control-estimation duality [100] establishes a correspondence between the problems of optimal control and optimal state estimation by highlighting the structural similarity between the design of a controller for a dynamical system and the design of an estimator for the system's state. Rather than producing 'new' information, dualities in control theory often are alternative representations of the same data (thus necessarily involutive) that are useful to identify these similarities, and potentially transport solution methods from one area to another.

In particular, the similarity between the computation of optimal costs of LQRs and optimal covariances for Kalman filters is first reflected by the fact that both covariance matrices and quadratic cost functions can be encoded as positive semi-definite matrices. This leads to the realization that the Kalman filter and the linear quadratic regulator are dual problems. Specifically, both problems are solved using a Riccati difference equation, but in dual forms: the Kalman filter uses the Riccati equation to propagate the estimation error covariance forward in time, whereas the LQR uses it to propagate the cost-to-go (value function) backward in time.

While for time-invariant systems, a duality entails a correspondence between the monoids of forward and backward Riccati equations, for time-variant systems, the duality must account for the time-dependent nature of the state and control spaces. This requires extending the duality to categories of controlled and observed processes, which we give in this section.

Recall how processes with control and observation were modelled as biparametrised endomorphisms in $\mathbf{Bipara}_+(\mathbf{FVec})$ (example 2.5.11). The dual space construction is

an adjunction $\mathrm{Hom}(-,\mathbb{R}) : \mathbf{Vect}_{\mathbb{R}}^{\mathsf{op}} \leftrightarrows \mathbf{Vect}_{\mathbb{R}} : \mathrm{Hom}(-,\mathbb{R})$ that restricts to an adjoint equivalence for the category of *finite dimensional* vector spaces $\mathbf{FVec}_{\mathbb{R}}^{\mathsf{op}} \cong \mathbf{FVec}_k$ [115, §IV.2]. We extend $\mathrm{Hom}(-,\mathbb{R}) \equiv (-)^\top$ to a homonymous functor that maps morphisms in $\mathbf{Bipara}_+(\mathbf{FVec})$ to their duals, while preserving the time-indexed structure. This duality will ensure that the forward and backward Riccati equations for time-variant systems are consistent with their time-invariant counterparts when restricted to fixed state and control spaces.

Definition 4.4.11 (Biparametrised duality)**.** The duality functor $(-)^*$ is an identity-on-objects, involutive contravariant endofunctor

$$(-)^* : \mathbf{Bipara}_+(\mathbf{FVec}_k)^{\mathsf{op}} \to \mathbf{Bipara}_+(\mathbf{FVec}_k) \tag{4.30}$$

with $k \in \{\mathbb{R}, \mathbb{C}\}$, whose definition on morphisms is given by transpose (resp. complex conjugation) and a permutation between parameters and coparameters. For $f : X \to Y$ given by (U, V, A, B, C, D), $f^* : Y \to X$ is given by $(V, U, A^*, C^*, B^*, D^*)$.

The definition of $(-)^*$ specialises to a functor relating observed and controlled processes via the projections $\mathbf{Bipara}_+(\mathbf{FVec}_k) \to \mathbf{Para}_+(\mathbf{FVec}_k)$ and $\mathbf{Bipara}_+(\mathbf{FVec}_k) \to \mathbf{Copara}_+(\mathbf{FVec}_k)$ defined in proposition 2.5.10 and the embeddings $\mathbf{Para}_+(\mathbf{FVec}_k) \to \mathbf{LinSpan}_k$ and $\mathbf{Copara}_+(\mathbf{FVec}_k) \to \mathbf{LinCospan}_k$ defined in proposition 2.5.6:

$$\begin{array}{ccc}
\mathbf{Bipara}_+(\mathbf{FVec}_k)^{\mathsf{op}} & \xrightarrow{(-)^*} & \mathbf{Bipara}_+(\mathbf{FVec}_k) \\
\downarrow & & \downarrow \\
\mathbf{Para}_+(\mathbf{FVec}_k)^{\mathsf{op}} & \xrightarrow{(-)^*} & \mathbf{Copara}_+(\mathbf{FVec}_k) \\
\downarrow & & \downarrow \\
\mathbf{LinSpan}_k^{\mathsf{op}} & \longrightarrow & \mathbf{LinCospan}_k
\end{array}$$

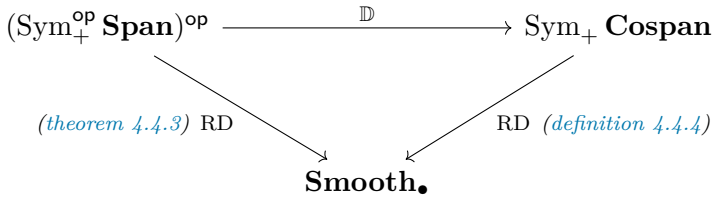
Moreover, the duality functor $(-)^*$ extends to a functor between the categories $\mathrm{Sym}_+^{\mathsf{op}}\,\mathbf{Span}$ and $\mathrm{Sym}_+\,\mathbf{Cospan}$, allowing us to relate the optimal control and estimation problems in a compositional way.

Definition 4.4.12 (Control-estimation duality functor)**.** The functor $\mathbb{D} : (\mathrm{Sym}_+^{\mathsf{op}}\,\mathbf{Span})^{\mathsf{op}} \to \mathrm{Sym}_+\,\mathbf{Cospan}$ is given by $(-)^* : \mathbf{LinSpan}_{\mathbb{R}} \to \mathbf{LinCospan}_{\mathbb{R}}$ on the base category, and

identity on the decorations. Concretely, it maps a morphism $(f, g, P) : n \leftarrow n+u \rightarrow m$ in $\mathrm{Sym}_+^{\mathsf{op}}$ **Span** to $(g^*, f^*, P) : m \rightarrow n+u \leftarrow n$ in Sym_+ **Cospan**.

This provides a unified framework for analysing the control-estimation duality in both time-invariant and time-variant settings.

Theorem 4.4.13. *The control-estimation duality functor $\mathbb{D}$ makes the following diagram commute; it gives a bijection between solutions of LQRs and solutions of EKFs.*

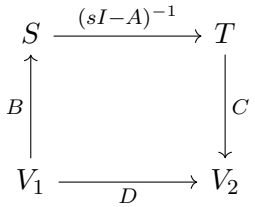


Proof. Recall that LQR maps are those in $\mathrm{Sym}_+^{\mathsf{op}}$ **Span** whose left leg is a projection, $n \xleftarrow{\pi_n} n+u \xrightarrow{f} m$, decorated by the immediate cost function $P_{n+u} \in \mathrm{Sym}_{\bullet+}(n+u)$. $\mathbb{D}(\pi_n, f, P_{n+u})$ is a morphism $m \rightarrow n$ in Sym_+ **Cospan** given by $m \xrightarrow{f^*} n+u \xleftarrow{\pi_n^*} n$. Note that because objects in these categories are *free* finite dimensional vector spaces, we identify the extension of functionals $\pi_n^* : (\mathbb{R}^n)^* \rightarrow (\mathbb{R}^{n+u})^*$ with the embedding of subspaces $\iota_n : \mathbb{R}^n \rightarrow \mathbb{R}^{n+u}$. Therefore, $\mathbb{D}(\pi_n, f, P_{n+u})$ can be identified with $m \xrightarrow{f^*} n+u \xleftarrow{\iota_n} n$ with decoration P_{n+u}, which is a symmetric square matrix over $\mathbb{R}^{n+u}$ that defines the correlation matrix of an extended Kalman filter. □

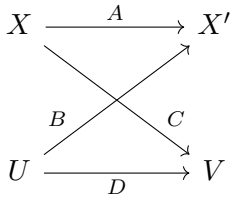
Putting aside the differences between the biparametrisation and prop approaches to control, the duality functor $(-)^*$ resembles Erbele's $-^*$ duality of the prop $\mathbf{ContFlow}_k$ [55]. One of the differences lies in the time reversal nature of $\mathbb{D}$, which in our first-class treatment of time steps—as domain and codomain of morphisms—is evident from its contravariance, while it remains hidden in the Laplace domain of Erbele. This Laplace transform reveals another key difference, which is the choice of interfaces and their composition (fig. 4.1).

4.4.5 Bellman lenses

Here we will relate the results in this chapter back to the lenses from chapter 3.



(a) Morphism in the prop **Stateful**$_k$ [55] representing the (Laplace transform of) a continuous dynamical system.



(b) Morphism in $\mathbf{Bipara}_+(\mathbf{FVec}_k)$ representing a discrete dynamical system.

Figure 4.1: Comparison between state space models in the prop and biparametrisation approaches.

At the end of example 4.4.10, we mention that the open-loop system $n_i \overset{\pi_{n_i}}{\leftarrow} n_i+u_i \overset{f_i}{\to} n_{i+1}$ with $f_i(\boldsymbol{x},\boldsymbol{u}) = A_i\boldsymbol{x} + B_i\boldsymbol{u}$, closed with the optimal control law $K_i^* : n_i \to u_i$, defines the optimal trajectory in **FVec**. Concretely, this optimal trajectory consists of linear maps $n_i \to n_{i+1}$ that take $\boldsymbol{x} \in \mathbb{R}^{n_i}$ to $(A_i + B_iK_i^*)(\boldsymbol{x}) \in \mathbb{R}^{n_{i+1}}$.

A similar construction derives optimal control laws for discounted costs, which introduces a discount factor $0 < \gamma < 1$ in the constraint pullback step as $P^*_{n_i+u_i} = P_{n_i+u_i} + \gamma f_i^\top P_{n_{i+1}} f_i$.

Let $P_{n_i+u_i}$ and $P_{n_{i+1}}$ be some fixed immediate and final costs functions. These determine K_i^*, which in the context of (time-discretized) LQRs as MDPs is realized as the *value improvement* optic $\mathbb{B}(K_i^*) : \binom{n_i}{1} \to \binom{n_{i+1}}{1}$ from the diset $\binom{n_i}{1}$—associated to the state space $\mathbb{R}^{n_i}$ at step i and the reward space $\mathbb{R}^1$—to the state and reward spaces at the next step $\binom{n_{i+1}}{1}$ in **Optic**(**Smooth**). Figure 4.2 shows its string diagram.

Perhaps more surprisingly, the functor RD : $(\mathrm{Sym}^{\mathsf{op}}_+ \mathbf{Span})^{\mathsf{op}} \to \mathbf{Smooth}_\bullet$ gives a characterization of the Bellman operator for *value iteration* (definition 3.3.2), that is, the application of the value improvement step followed by the policy improvement step that *computes* the optimal control laws, for linear dynamics and quadratic costs/utilities. Roughly, the constraint pullback step corresponds to value improvement and the Schur complement to policy improvement. This addresses the efficiency issue of the policy improvement map in (3.23) (see remark 3.5.4), which when applied to the LQR problem either iterates through all control vectors in $\mathbb{R}^u$ (by exhaustive search of the set with cardinality $|\mathbb{R}^u|$) or searches for a free linear map $n \to u$ (in a space with

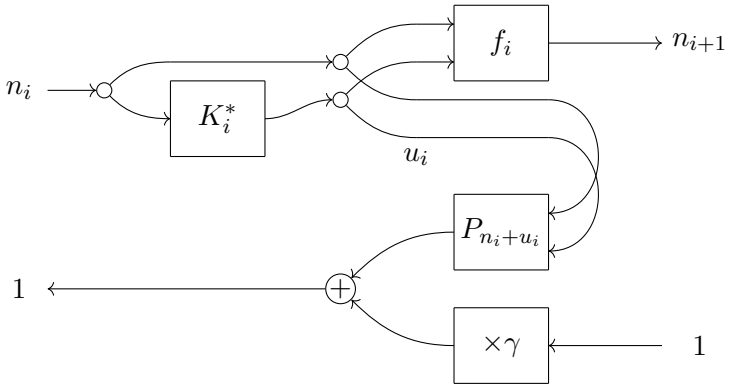


Figure 4.2: Value improvement optic for the optimal control law in a linear quadratic regulator.

cardinality $n \times u$) which ignores any linear map that is not freely generated from its function representation in **Set**.

This cannot be stated as a lens or mixed optic (without stretching their definitions too much), as their usual formulations require the forwards map to be a morphism in a 'directional' category (section 1.1). If we were to propose an optic that resembles (4.22) with type $\binom{n}{n(n+1)/2} \to \binom{m}{m(m+1)/2}$ in $\mathbf{Optic}(\mathbf{FVec}_{\mathbb{R}}, \mathbf{Smooth})$, the definition of such a putative mixed optic would be given by $\int^{u:\mathbf{FVec}} \mathbf{FVec}(n, u \bullet m) \times \mathbf{Smooth}(u \bullet m(m+1)/2, n(n+1)/2)$, which already *requires* the pre-computation of a policy or control law (optimal or not) to define a forwards map $n \to u + m$, whereas finding this control law is the whole point of studying this problem.

Chapter 5

Reinforcement Learning

5.1 Introduction

Reinforcement Learning (RL) may be defined as optimal control of agents that don't have perfect knowledge of their environment. The modern landscape of this field is often characterized by a rich yet fragmented 'algorithmic zoo', where subtle variations in environment representation, objective functions, and optimization methods obscure fundamental conceptual similarities. Several theoretical frameworks offer unifying perspectives, usually based on statistical methods, information theory, and ODEs [158, 23, 24, 120]. The core motivation for this chapter is to build on the lessons on Category Theory applied to Dynamic Programming learned in chapter 3, to provide some structural insights into the field of RL.

By defining Bellman equations, policies, value functions, and the decision-making process within categorical structures, we aim to hopefully pave some of the way towards the seemingly distant goal of enabling systematic comparison, generalization, and the principled derivation of novel learning algorithms from first principles, ultimately establishing a more robust and coherent mathematical foundation for the field.

Chapter 3 shows that *value iteration*, a fundamental method common to both dynamic programming and reinforcement learning, 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[1]. This has the property that for any *value function* $V : S \to \mathbb{R}$, represented as an optic $V : \binom{S}{\mathbb{R}} \to I$, the composed $V \circ \mathbb{B}(\pi)$ is a better value function.

As the optic formalisation of these operators worked for perfect knowledge of the environment, we now take this idea to express other algorithms in reinforcement learning where agents only learn from the environment by interacting with it through samples. The outline of the main construction in this chapter is: (1) We extend $\mathbb{B}(\pi)$ to a *parametrised optic* representing a more general class of Bellman operators known as *empirical Bellman operators* that apply to action-value functions and depend on a *sample* as a parameter. (2) We apply the continuation functor $\mathbb{K}$, a representable contravariant functor that already plays a foundational role in compositional game theory, obtaining a parametrised function $\mathfrak{B} = \mathbb{K}(\mathbb{B}(\pi))$ 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 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 (see e.g. [2]). An earlier version of this chapter can be found at [84].

5.2 When models are not perfect

Because of its interdisciplinary origins, the fragmentation of algorithm families in RL also affects its terminology. Many fundamental concepts—such as agent, environment or model—are often inconsistent, so we choose to introduce them in this section oper-

[1] In this chapter, we write S for the state space instead of X following convention in the RL literature. They can be treated interchangeably.

ationally rather than through formal propositions. Subsequent sections will introduce more concrete definitions for some of the terms once the sufficient theory is developed.

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.

This thesis adopts a structural approach to the study of these algorithms, with the main structural distinction drawn between the agent and the environment. The **environment** represents the external system with which the agent interacts. It can have a lot of structure—from non-linear dynamical systems to non-deterministic IO—but as agents only interact with it through samples, it may be assumed to be a Markov decision process. A Markov process consists of a set of states S and a stochastic transition function $f : S \to \mathrm{D}S$, where D is some probability monad over **Set** and f is given by probabilities $f(s' \mid s)$. A Markov reward process is a Markov process with an additional function $r : S \to \mathrm{D}\mathbb{R}$ that outputs the immediate *reward* for the current state. This reward function can be in general be correlated with the transition, in which case we write $f : S \to \mathrm{D}(S \times \mathbb{R})$. When clear from context, let r stand for the reward of a particular state, e.g. $(s', r) \sim f(s)$.

Markov decision processes (MDP), which we introduced in section 3.2, are MRPs with a set A of actions, whose transition and reward functions now depend additionally on the action taken at each state, and is also in general correlated as well $f : S \times A \to \mathrm{D}(S \times \mathbb{R})$. The cases where the reward function is decorrelated with f as in $S \times A \to \mathrm{D}\mathbb{R}$, $S \to \mathrm{D}\mathbb{R}$, or $S \times A \times S \to \mathrm{D}\mathbb{R}$ can be embedded in our modelling choice for f.

As in Dynamic Programming (DP), an agent's goal is to maximize the *expected long-run reward* $\sum \gamma^i r(s_i)$ (cf. (3.2)), where $0 < \gamma \leq 1$ is the **discount factor**, a hyperparameter that controls the agent's "patience", or preference between rewards in the present and rewards in the future.

Unlike DP however, the agent's knowledge about the environment consists of the MDP's interface via S, A and $\mathbb{R}$, but not the definition of the transition nor reward functions. The environment's response to an action of the agent, described next, is given by the transition dynamics f that can be assumed to have the Markov property, and the environment's state is known to the agent. The agent learns therefore only via *samples*; if we assume the environment's transition function is given by $f : S \times A \to \mathrm{D}(S \times \mathbb{R})$, the agent only has access to points in $S \times A \times S \times \mathbb{R}$. When the agent's knowledge of the environment is further limited to partial observations of the state S, the setting is modelled as a partially observable MDP (POMDP), and samples are points of $O \times A \times O \times \mathbb{R}$, where $\iota : O \hookrightarrow S$ is unknown too.

The core components of an **agent** are the policy, the reward, the value function and the internal model. A **policy** $\pi : S \to TA$ defines its strategy mapping states to actions, for a monad T associated to the type of policy. 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 ($T = \mathrm{D}$ the same distribution functor as the environment, like ε-greedy), with probabilistic being the most common. 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. 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* [147]. A **value function** estimates this expected long-term reward associated with following a particular policy. Usually one works with either a state value function $V : S \to \mathbb{R}$ or a state-action value function $Q : S \times A \to \mathbb{R}$, 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.

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 a (possibly partial) 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**. **On-policy** methods only have a single 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.

The diversity of methods encompassed by RL employs experimental and formal justifications to tackle weak spots in this learning theory such as the credit-assignment problem, the exploration-exploitation trade-off and coping with state that is hidden or too big to represent explicitly. Notably, these challenges were also recognized in earlier disciplines, including psychology and neuroscience [98].

A central class of approaches within RL is the family of DP algorithms, discussed in detail in chapter 3. These methods are an idealized class of model-based algorithms that do not need to interact with the environment because they rely on a perfect model of it as an MDP. The search for an optimal policy happens therefore entirely within the agent's model, interleaving the two feedback operations—*value improvement* and *policy improvement*—in a process of updating previous estimates called **bootstrapping**.

Beyond dynamic programming, RL includes additional algorithmic paradigms such as Monte Carlo methods, temporal difference learning, and deep reinforcement learning. An overview of each will be given in subsequent sections.

5.2.1 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 an 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 $\pi : S \to \mathrm{D}A$ is derived: $\pi(s) = \operatorname{argmax}_a Q(s, a)$ 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), we define the update **target**—denoted in the rest of this chapter as G, following [158]—to be $G = \sum_t \gamma^t r_t$, and the value function updates as

$$Q'(s, a) = (1 - \alpha)Q(s, a) + \alpha G \tag{5.1}$$

where the learning rate $0 \leq \alpha \leq 1$ 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.

5.2.2 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 (5.1) 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 5.2.1 (SARSA [157])**.** 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 $G = r + \gamma \mathbb{E}_{a \sim \pi_{\text{tgt}}(s)} Q(s', a')$, which is off-policy because the last action is determined in expectation by a target policy π_{tgt}.

Example 5.2.2 (Q-learning [166])**.** In Q-learning, given the current state s the agent performs an action $a \sim \pi_{\text{beh}}(s)$ 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

$$G = r + \gamma Q(s', \pi_{\text{tgt}}(s')) = r + \gamma \max_{a' \in A} Q(s', a') \tag{5.2}$$

It is an off-policy method because the last action used to compute the update is $\pi_{\text{tgt}}(s') = \text{argmax}_{a' \in A} Q(s', a')$ and not $\pi_{\text{beh}}(s')$.

Note that both the new state s' and the reward r are obtained by interacting with the system, rather than looked ahead by $s' = f(s, a)$ and $r = U(s, a)$, again because the agent does not have knowledge about the environment's dynamics. This illustrates the important distinction in RL 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 $\text{argmax}_{a' \in A} Q(s', a')$ is used only when computing updates. We can consider this to be a separate target policy, $\pi_{\text{tgt}}(s') = \text{argmax}_{a' \in A} Q(s', a')$. The reader may note from (5.2) the resemblance to Exp-SARSA: one way to define Q-learning is Exp-SARSA with a greedy target policy.

5.2.3 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 (see section 3.5.2). In the context of RL, the family of supervised-learning function approximation methods are known as *deep reinforcement learning*, based on the same **gradient-based** methods as deep learning.

Given a loss function $\mathcal{L} : \Theta \to \mathbb{R}$ from parameter space Θ to real values, these methods minimise $\mathcal{L}$ by iteratively updating the model parameters θ in the direction that reduces the loss, that is, the opposite to the gradient $\nabla_\theta \mathcal{L} : T_\theta \Theta$. In contrast

to traditional machine learning methods where this loss function is computed from an error function (in supervised learning like classification or regression) or from data-intrinsic objectives reflecting structural fit (in unsupervised learning like k-means used for clustering), RL computes loss from errors against bootstrapped value estimates derived from rewards and prior predictions.

By trading the computational burden of calculating the Jacobian matrix from the entire dataset to the lower convergence of estimating it from a sample (x_i, y_i) or subset $\{(x_i, y_i)\}_i^n$ of samples [35], **stochastic gradient descent** (SGD) [137, 73, 24] approximates the parameter update

$$\theta' = \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$$

computed from the whole dataset with an update computed from a sample (x_i, y_i):

$$\theta' = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, x_i, y_i) \tag{5.3}$$

Deep Q-networks (DQN) [122] 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 stochastic (semi)gradient[2] descent 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 (5.1) becomes $Q'(s, a) = Q(s, a) - \frac{\alpha}{2} \partial_{Q(s,a)} \mathcal{L}(Q(s, a), G)$. Therefore, **Bellman updates are SGD on points**. 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 SGD and other 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 [159] bypass the generation of a value altogether, by directly outputting an action distribution $\mathrm{PG} : \Theta \times S \to \mathrm{D}A$. One can see DQNs

[2] Semi-gradient because the Q-function is contained in the target, yet it is considered constant in the gradient computation [158, §9.3][16].

as a special PG method by transforming the Q-function into an action probability via the Boltzmann distribution, also known as softmax. The policy gradient theorem [117] states that the gradient of the expected cumulative reward with respect to model parameters $\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta} \sum_{t=1}^{T} r_t$ does not depend on

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{\tilde{\pi}_\theta}\left[\sum_{t=1}^{T} r_t \cdot \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

This justifies the lack of model of certain methods like REINFORCE. **Actor-Critic** (AC) methods [107] are a further improvement over this class (see example 5.5.1), solving the high sample variance by introducing a baseline value function learned by a DQN called the *critic*. The PG-like network in AC is called the *actor*, hence the method name.

5.3 States, contexts and iteration

An optic, as introduced in section 2.5.4 (definition 2.5.13), is a process consisting of a forward pass followed by a backward pass. In many applications 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. This section will develop a theory of iterated optics, partly popularized by [78].

Definition 5.3.1 (Iteration functor)**.** Let $\mathcal{C}$ be a symmetric monoidal category. The *iteration functor* is a symmetric lax monoidal functor $\mathbb{I} : \mathbf{Optic}(\mathcal{C}) \to \mathbf{Set}$. On objects, we set

$$\mathbb{I}\binom{X}{X'} = \int^{M:\mathcal{C}} \mathcal{C}(I, M \otimes X) \times \mathcal{C}(M \otimes X', M \otimes X)$$

Given a representative 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 $\mathbf{Optic}(\mathcal{C})$, 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

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

Proposition 5.3.2. *The iterator* $\mathbb{I} : \mathbf{Optic}(\mathcal{C}) \to \mathbf{Set}$ *is well-defined.*

Proof. We have to prove that the following function is well-defined:

$$\mathbf{Optic}\left(\binom{X}{X'}, \binom{Y}{Y'}\right) \to \left[\mathbb{I}\binom{X}{X'} \to \mathbb{I}\binom{Y}{Y'}\right] \tag{5.4}$$

Following Riley's proof method of sequential composition of optics [136], the uncurried form of the above function has as domain:

$$\begin{aligned}
&\left(\int^{M} \mathcal{C}(X, M \otimes Y) \times \mathcal{C}(M \otimes Y', X')\right) \times \left(\int^{N} \mathcal{C}(I, N \otimes X) \times \mathcal{C}(N \otimes X', N \otimes X)\right) \\
\cong &\int^{M,N} \mathcal{C}(X, M \otimes Y) \times \mathcal{C}(M \otimes Y', X') \times \mathcal{C}(I, N \otimes X) \times \mathcal{C}(N \otimes X', N \otimes X) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(coend-Fubini)}
\end{aligned}$$

By the universal property of coends it suffices to construct maps natural in M and N into the codomain of (5.4):

$$\begin{aligned}
&\mathcal{C}(X, M \otimes Y) \times \mathcal{C}(M \otimes Y', X') \times \mathcal{C}(I, N \otimes X) \times \mathcal{C}(N \otimes X', N \otimes X) \\
\to &\mathcal{C}(I, N \otimes M \otimes Y) \times \mathcal{C}(N \otimes M \otimes Y', N \otimes M \otimes Y) && \text{(composition in } \mathcal{C}) \\
\to &\int^{P} \mathcal{C}(I, P \otimes X) \times \mathcal{C}(P \otimes X', P \otimes X) && (\mathrm{copr}_{N \otimes M})
\end{aligned}$$

where the first map takes morphisms f, f', x_0, i to

$$\begin{aligned}
&I \xrightarrow{x_0} N \otimes X \xrightarrow{N \otimes f} N \otimes M \otimes Y \\
&N \otimes M \otimes Y' \xrightarrow{N \otimes f'} N \otimes X \xrightarrow{i} N \otimes X \xrightarrow{N \otimes f} N \otimes M \otimes Y
\end{aligned}$$

This also admits a graphical representation, depicted in fig. 5.1. □

Proposition 5.3.3. *The iterator* $\mathbb{I} : \mathbf{Optic}(\mathcal{C}) \to \mathbf{Set}$ *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

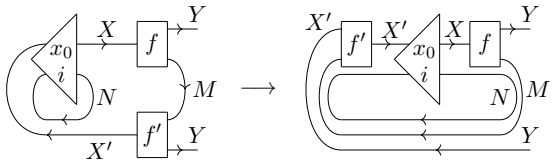


Figure 5.1: Composition of an optic with an iterator yields another iterator.

morphisms in $\mathbf{Optic}(\mathcal{C})$. 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

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

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 $(M \otimes (N \otimes g'); f'); i; (M \otimes (f; N \otimes g))$. □

When $\mathcal{C} = \mathbf{Set}$ 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 $\langle -|-\rangle : \mathbb{K}\binom{X}{X'} \times \mathbb{I}\binom{X}{X'} \to X^\omega$ which is well-defined.

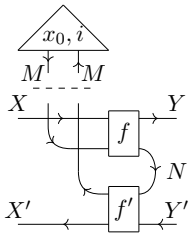


Figure 5.2: Typical morphism in $\mathbf{Optic}^{\mathbb{I}}(\mathcal{C}) = \pi_0^* \left(\mathbf{Para}^{\mathbb{I}}(\mathbf{Optic}(\mathcal{C}))\right)$ which consists of a morphism in $\mathbf{Optic}(\mathcal{C})$ extended by a representative element $(M, x_0, i) \in \mathbb{I}\binom{X}{X}$. Adapted from [78].

Proposition 5.3.4. *The map* $\langle - \mid - \rangle : \mathbb{K}\binom{X}{X'} \times \mathbb{I}\binom{X}{X'} \to X^\omega$ *is a dinatural transformation when* $\mathcal{C} = \mathbf{Set}$.

Proof. Considering that $\mathbf{Optic}(\mathbf{Set}) \cong \mathbf{Lens}$, the domain and codomain functors of the transformation are $\mathbb{K} \times \mathbb{I} : \mathbf{Lens}^{\mathsf{op}} \times \mathbf{Lens} \to \mathbf{Set}$ and $\mathbb{V}^\omega : \mathbf{Lens} \to \mathbf{Set}$, where $\mathbb{V}^\omega$ is the forwards pass functor $\mathbb{V} : \mathbf{Lens} \to \mathbf{Set}$ followed by the stream functor $(-)^\omega : \mathbf{Set} \to \mathbf{Set}$. This being a purely covariant functor makes the dinaturality condition into the following pentagon identity for every lens $\lambda : \binom{X}{X'} \to \binom{Y}{Y'}$ whose forward and backward maps we denote f and f':

$$\begin{array}{ccccc}
 & & \mathbb{K}\binom{X}{X'} \times \mathbb{I}\binom{X}{X'} & \xrightarrow{\langle - \mid - \rangle_{\binom{X}{X'}}} & \mathbb{V}^\omega\binom{X}{X'} = X^\omega \\
 & \nearrow^{\mathbb{K}(\lambda) \times \mathbb{I}\binom{X}{X'}} & & & \\
\mathbb{K}\binom{X}{X'} \times \mathbb{I}\binom{X}{X'} & & & & \Big\downarrow {\scriptstyle \mathbb{V}^\omega(\lambda) = f^\omega} \\
 & \searrow_{\mathbb{K}\binom{X}{X'} \times \mathbb{I}(\lambda)} & & & \\
 & & \mathbb{K}\binom{X}{X'} \times \mathbb{I}\binom{X}{X'} & \xrightarrow{\langle - \mid - \rangle_{\binom{X}{X'}}} & \mathbb{V}^\omega\binom{X}{X'} = Y^\omega
\end{array}$$

For $k, (M, (m_0, x_0), i)$ in $\mathbb{K}\binom{Y}{Y'} \times \mathbb{I}\binom{X}{X'}$, the diagram commutes when the streams $f^\omega \langle \lambda; k \mid M, (m_0, x_0), i \rangle$ and $\langle k \mid M, (m_0, f(x_0)), j \rangle$ are equal, where j is given by

$$j = (M \times f'); i; (M \times f) \tag{5.5}$$

We proceed by coinduction, showing that the streams have equal heads and tails; an equivalent perspective is that they are generated by bisimilar (in fact isomorphic) state machines. The heads of both streams is $f(x_0)$, and their tails operate as state machines for states (m_0, x_0):

$$f^\omega \langle \lambda; k \mid M, i(m_0, (\lambda; k)(x_0)), i\rangle = f(i(m_0, (\lambda; k)(x_0))_1) :: \langle \cdots \rangle$$
$$\langle k \mid M, j(m_0, k(f(x_0))), j\rangle = j(m_0, k(f(x_0)))_1 :: \langle \cdots \rangle$$

Thus we show for all pairs (m, x),

$$\begin{aligned}
& f(i(m, (\lambda; k)(x))_1) \\
=& f(i(m, (f; k; f')(x))_1) && \text{(Composition of lens with continuation)} \\
=& ((M \times f'); i; (M \times f))(m, k(f(x)))_1 \\
=& j(m, k(f(x)))_1
\end{aligned}$$

□

In the general case, we believe this can be accomplished using the machinery of monoidal streams [111].

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 $\mathbf{Optic}^{\mathbb{I}}(\mathcal{C}) = \pi_0^* \left(\mathbf{Para}^{\mathbb{I}}(\mathbf{Optic}(\mathcal{C}))\right)$ extends $\mathbf{Optic}(\mathcal{C})$ with states $I \to \binom{X}{X'}$ defined by elements of $\mathbb{I}\binom{X}{X'}$. A typical morphism is depicted in fig. 5.2.

Given a symmetric monoidal category $\mathcal{C}$, a **context** for morphisms $X \to X'$ is a state $I \to \binom{X}{X'}$ of $\mathbf{Optic}(\mathcal{C})$. When $\mathcal{C}$ is itself a category of optics, this is known as **double optics** and is a central idea of Bayesian open games [27]. These can be depicted as combs with one hole and bidirectional wires, or combs with two holes and only

forwards wires.

Putting this together, we are now in a position to define *iteration contexts* for optics $\binom{X}{X'} \to \binom{Y}{Y'}$.

Definition 5.3.5. An **iteration context** for the category $\mathbf{Optic}^I(\mathcal{C})$ is a (representable) state of $\mathbf{Optic}(\mathbf{Optic}^{\mathbb{I}}(\mathcal{C}))$. This defines a functor $\mathbb{I}_{\text{env}} : \mathbf{Optic}^2(\mathcal{C}) \to \mathbf{Set}$ depicted in fig. 5.3(left), and by pulling the state variable of i through k we can define it equivalently by a coend over $\mathcal{C}$ rather than over $\mathbf{Optic}^{\mathbb{I}}(\mathcal{C})$:

$$\mathbb{I}_{\text{env}}\left(\binom{X}{X'}, \binom{Y}{Y'}\right) \cong \int^{M,M':\mathcal{C}} \mathcal{C}(I, M \otimes X) \times \mathcal{C}(M \otimes Y, M' \otimes Y') \times \mathcal{C}(M' \otimes X', M \otimes X) \tag{5.6}$$

The idea of iteration contexts can also be presented under alternative formalisations based on Tambara theory [131] and combs [138, §4.3][111][86].

Remark 5.3.6. Given a symmetric monoidal category $\mathcal{C}$, a functor $\mathbf{Optic}(\mathcal{C}) \to \mathbf{Set}$ can be equivalently defined as a **Tambara comodule**: a functor $W : \mathcal{C} \times \mathcal{C}^{\mathsf{op}} \to \mathbf{Set}$ (note the variance) 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 [131, 46]. Given this data, a 'generalised comorphism' $X \to X'$ can be defined as an element of $W\binom{X}{X'}$, that is, a state of $\mathbf{Optic}^W(\mathcal{C}) = \pi_0^*\left(\mathbf{Para}^W(\mathbf{Optic}(\mathcal{C}))\right)$ (see [81, §9]).

Remark 5.3.7 (Combs for iteration contexts)**.** An alternative depiction of iteration contexts in fig. 5.3(right) shows the coend (5.6) as a 3-hole comb. Moreover, the unrolling into n steps produces $2n$-hole combs, including ω-combs [111] for the limiting case.

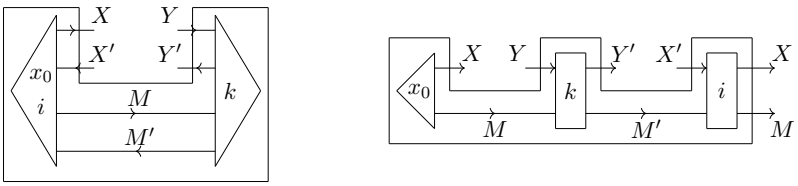


Figure 5.3: Iteration contexts as states of $\mathbf{Optic}^2(\mathcal{C})$ (left) and as 3-hole combs (right).

5.4 Empirical Bellman operators

The type of Bellman operator introduced in section 3.3 updates an entire value function at once, as is most common in basic dynamic programming. However, as discussed in section 5.2, reinforcement learning, viewed as 'incremental dynamic programming', updates the value function one entry at a time, with a sample that acts as a moving target determining which state-action pair must update its value [166].

This kind of Bellman operators do not directly factor through $\mathbb{K}$, and we propose an alternative factorisation that we sketch next.

Definition 5.4.1 (Empirical Bellman operator, cf. [158, 77])**.** Let $\mathfrak{B} : \mathbb{R}^{S\times A} \to \mathbb{R}^{S\times A}$ be a Bellman operator. An *empirical Bellman operator* is a map $\Upsilon \times \mathbb{R}^{S\times A} \to T(\mathbb{R}^{S\times A})$ that assigns to each sample in Υ and point $Q \in \mathbb{R}^{S\times A}$ a tangent vector in $T_Q\mathbb{R}^{S\times A}$.

The Bellman error or *target* is given by $\Upsilon \times \mathbb{R}^{S\times A} \to S \times A \times \mathbb{R}$, whose output (s, a, r) is the (s, a)-component of the tangent vector $T_Q\mathbb{R}^{S\times A}$. For a space of value functions Θ, let $\Delta\Theta$ denote the space of these components of tangent vectors. So $\Delta(\mathbb{R}^{S\times A}) \cong S \times A \times \mathbb{R}$, $\Delta(\mathbb{R}^S) \cong S \times \mathbb{R}$ and $\Delta(A^S) \cong S \times A$.

From this, we understand the empirical Bellman operator to be composed of $|S|\times|A|$ Bellman errors or 'deltas' that define all components of the true tangent vector. This same change of perspective applies to deep RL where the delta is replaced with a gradient vector, so it is an interesting observation that the category theory suggests the same for the more discrete setting of tabular RL. In the same approximation that SGD does of true gradient descent (recall section 5.2.3), Bellman errors provide update rules that in the limit approach the true Bellman operator.

Example 5.4.2. Consider an environment with three states and two actions. Assume that the agent has a Q-function $Q : 3\times 2 \to \mathbb{R}$ that estimates the value of all state-action value pairs. If the interaction with the environment returns samples $\Upsilon = S \times A \times S \times \mathbb{R}$ of a state, the action taken in that state, the next state and a reward value, a Bellman

error given by Q-learning (example 5.2.2) is:

$$\begin{aligned} &3 \times 2 \times 3 \times \mathbb{R} \times \mathbb{R}^{3\times 2} \to 3 \times 2 \times \mathbb{R} \\ &s, \quad a, \quad s', \quad r, \quad Q \quad \mapsto s, \quad a, \quad \max_{a' \in 2} Q(s', a') \end{aligned}$$

The bellman errors or targets computed from samples for all $|S| \times |A| = 6$ state-action pairs define a tangent vector $T_Q \mathbb{R}^{3\times 2}$.

Remark 5.4.3. As the state value and state-action value functions are elements of euclidean spaces $\mathbb{R}^S$ and $\mathbb{R}^{S\times A}$, we treat the tangent and cotangent spaces as isomorphic via their canonical Riemannian metric. It can be argued from the differential geometry perspective of Myers' categorical systems theory [125] and Capucci [40], that the correct definition of empirical Bellman operators, in analogy to gradient descent, should output *cotangent* rather than tangent vectors.

Definition 5.4.4 (Externally parametrised lenses)**.** Let $\mathbf{Lens} = \mathbf{Optic}(\mathbf{Set})$ be the category of lenses over sets. Since $\mathbf{Lens}$ is enriched in $\mathbf{Set}$, we can form its category of externally parametrised morphisms, $\mathbf{Para}_{\mathbf{Set}}(\mathbf{Lens})$. A morphism $\binom{X}{X'} \to \binom{Y}{Y'}$ of this category consists of:

i. a parameter set Υ (representing samples in the context of RL),

ii. a forwards pass function $\Upsilon \times X \to Y$,

iii. a backwards pass function $\Upsilon \times X \times Y' \to X'$.

Let us consider one of the Bellman operator for SARSA (example 5.2.1).

Example 5.4.5 (SARSA, continued)**.** The Bellman update for SARSA is a lens $\mathfrak{B} : \binom{1}{S\times A\times \mathbb{R}} \to \binom{S\times A}{\mathbb{R}}$, with parameter set $\Upsilon = S \times A \times \mathbb{R} \times S \times A$, where the backward pass function is $r, v \mapsto r + \gamma v$ (fig. 5.4).

We lift the functor $\mathbb{K} : \mathbf{Lens}^{\mathsf{op}} \to \mathbf{Set}$ to the functor $\mathbf{Para}_{\mathbf{Set}}(\mathbb{K}) : \mathbf{Para}_{\mathbf{Set}}(\mathbf{Lens}^{\mathsf{op}}) \to \mathbf{Para}_{\mathbf{Set}}(\mathbf{Set})$. Applying this functor to $\mathfrak{B}$ results in a Bellman error

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

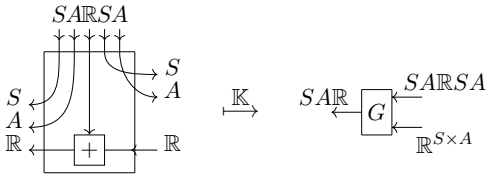


Figure 5.4: Target computation in SARSA as a parametrised lens, and its $\mathbf{Para}_{\mathbf{Set}}(\mathbb{K})$-image in **Set**.

where $S \times A \times \mathbb{R}$ is the Bellman error $\Delta(\mathbb{R}^{S\times A})$ at $Q \in \mathbb{R}^{S\times A}$. The application of this Bellman error in the update of a value function Q consists of the substitution of the value $Q(s,a)$ with the new $r + \gamma Q(s',a')$ or an interpolation between the old and new values weighted by a given learning rate $0 \leq \alpha \leq 1$. This defines an iterator (M, q_0, i) on $\begin{pmatrix} \mathbb{R}^{S\times A} \\ \Delta(\mathbb{R}^{S\times A}) \end{pmatrix}$ (definition 5.3.1), where $M = \mathbb{R}^{S\times A}$, $q_0 : I \to M \otimes \mathbb{R}^{S\times A}$ is the initial choice of Q-matrix Q, and the iterator $i : M \times \Delta(\mathbb{R}^{S\times A}) \to M \otimes \mathbb{R}^{S\times A}$ is given by:

$$\begin{aligned} i : \mathbb{R}^{S\times A} \times \Delta(\mathbb{R}^{S\times A}) &\to \mathbb{R}^{S\times A} \otimes \mathbb{R}^{S\times A} \\ Q, \quad s, a, r &\mapsto Q', Q' \end{aligned} \tag{5.7}$$

with $Q' : S \times A \to \mathbb{R}$ being:

$$\begin{aligned} Q' : S \times A &\to \mathbb{R} \\ (s',a') &\mapsto \begin{cases} r & \text{if } s = s', a = a' \\ Q(s',a') & \text{otherwise} \end{cases} \end{aligned}$$

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 $G : \Upsilon \times \mathbb{R}^S \to \Delta(\mathbb{R}^S)$ in **Set**. This morphism becomes a central part of our formalisation of an RL model, explained next.

5.5 Models, agents and environments

We understand a *model* for an RL method to contain the data to generate the policy from certain inner parameters and the data to update those parameters based on bootstrapping and/or samples via an empirical Bellman operator (definition 5.4.1). It matches the structure of a lens from model parameters to agent interface, which we annotate in fig. 5.5a.

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 (fig. 5.5b), $P : \mathbb{R}^{S\times A} \to (\mathrm{D}A)^S$ takes the current Q-table $Q : \mathbb{R}^{S\times A}$ and returns the greedy policy $\pi : (\mathrm{D}A)^S$ defined by $\pi(s) = \mathrm{argmax}_a\, Q(s,a)$.

The backward map takes the sample returned from the agent and the current parameters to generate an update target as a Bellman error for the parameters. The sample space Υ usually takes the form of some product of types S (states), A (actions) and $\mathbb{R}$ (rewards). In SARSA (fig. 5.5b), 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 tangent at (s, a).

Non-bootstrapping methods correspond to optics whose action is given by the terminal category (called *isos* or *adaptors* in the optics literature [132][136, §4.3]). For instance, the backward map of MC (fig. 5.5c) calculates the target without bootstrapping the value function $Q : \mathbb{R}^{S\times A}$.

In DP (fig. 5.5d), 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 tangent 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 (definition 3.3.3) or asynchronous DP [21] don't enforce an execution order for value improvement and policy improvement, and benefit from treating the two Bellman operators as separate backward morphisms (fig. 5.5e).

This also captures some deep reinforcement learning methods, which use a gradient-based approximation of the value functions instead of a tabular approach.

Example 5.5.1 (Actor-critic)**.** Actor-critic methods [107] consist of a neural network

policy called the *actor*, and an additional function called the *critic* that learns a baseline value function. The actor $P : \Theta \to (S \to DA)$ appears in the forward map of the diagram (fig. 5.5f), but the critic $V : \Omega \to (S \to \mathbb{R})$ does not, as it does not act on the environment. Being a deep RL method, the actor map $P : \theta \mapsto \pi_\theta$ is a neural network, and the associated backward loss map $\mathcal{L}_{\text{ac}}$ is defined by the improvement of expected return

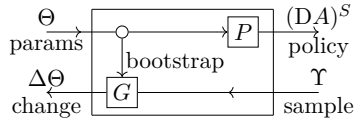
$$\begin{aligned} \mathcal{L}_{\text{ac}} : \Theta \times \Omega \times SA\mathbb{R}S &\to \Delta\Theta \\ (\theta, \omega, s, a, r, \_) &\mapsto (r - V_\omega(s))\nabla_\theta \log \pi_\theta(s, a) \end{aligned}$$

with the baseline given by V_ω. The critic map $V : \omega \mapsto V_\omega$ has as the backward loss map $\mathcal{L}_{\text{cr}}$ the reduction of policy update variance

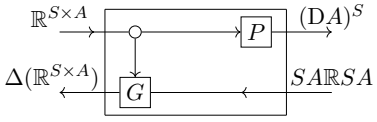
$$\begin{aligned} \mathcal{L}_{\text{cr}} : \Omega \times SA\mathbb{R}S &\to \Delta\Omega \\ (\omega, s, a, r, s') &\mapsto r + \gamma V_\omega(s') - V_\omega(s) \end{aligned}$$

This generic model lens, and any of the variants above, embed into $\mathbf{Optic}(\mathcal{C})$ which is extended to $\mathbf{Optic}^{\mathbb{I}}(\mathcal{C})$ by an iteration functor $\mathbb{I}$ that we illustrated for SARSA in (5.7). 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. The bootstrapping type M is usually the whole parameter space Θ, but the optic iteration functor allows M to be any state space, e.g. a subset of Θ or an alternative encoding of it. In gradient-free methods, the update is generally pointwise as in (5.1). Conversely, gradient-based methods use neural network optimizers like stochastic gradient descent, Adam and other variations as the update rule. However, as mentioned in section 5.2.3, the change value passed to the update rule is computed from bootstrapped values and sample trajectories from the environment, and do not come directly from an error function with respect to labelled data, as done for example for lens semantics of SGD in [48, §3.3] for supervised learning.

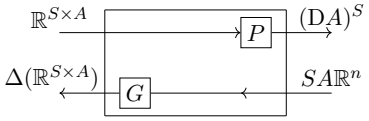
The right interface of a model parametrises an agent, which is a morphism $(N, \text{ev}, \text{ret}) : \binom{S}{I} \to \binom{A}{F}$ in $\mathbf{Para}(\mathbf{Optic}(\mathcal{C}))$ for some $N : \mathcal{C}$. The forward map $\text{ev} : S \otimes (\text{D}A)^S \to$



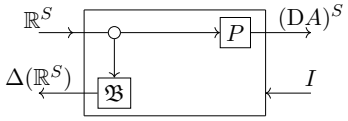
(a) Generic RL



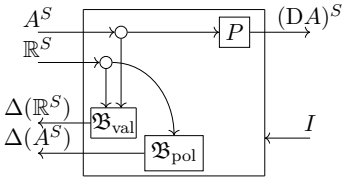
(b) Temporal difference (TD)



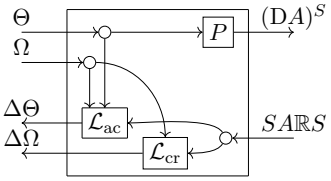
(c) Monte Carlo (MC)



(d) Dynamic programming (DP)



(e) Generalized policy iteration (GPI)



(f) Actor-critic (AC)

Figure 5.5: Lens string diagrams for several RL model architectures (recall fig. 2.3).

$A \otimes N$ evaluates a given policy $\pi : (\mathrm{D}A)^S$ in a state $s : S$, resulting in an action $a : \pi(s)$ and some internal state $n : N$. Note that the evaluation map is not canonical since a Markov category is not cartesian closed in general. The backward map $\mathrm{ret} : N \otimes F \to \Upsilon$ aggregates any internal state $n : N$ and the feedback from the environment F to generate a sample in Υ that goes back to the model as a coparameter. As it is common that the model uses any possible information from the agent's interaction with the environment, this map is usually an isomorphism $N \otimes F \cong \Upsilon$.

The coend in the environment is taken over states of the Markov chain. Even though samples have been described so far as elements in the set Υ, these points can be equally be described as points of a random variable $\mathcal{C}(I, \Upsilon)$ in the ambient Markov category $\mathcal{C}$. Moreover, the random variable is correlated with any effect that the agent has on the environment. This is illustrated in the next proposition (whose string diagram representation is shown in fig. 5.6), were the composition of an agent (a parametrised optic) with an environment (an iteration context) is described in detail for the case of

Q-learning.

Proposition 5.5.2 (Agent-environment interation in Q-learning)**.** *Let* $\mathbb{I}_{env}\left(\binom{S}{I}, \binom{A}{S'\otimes\mathbb{R}}\right)$ *be an environment in a Markov category* $\mathcal{C}$*, and* $(S\otimes A, \mathrm{ev}, 1) : \binom{S}{I} \to \binom{A}{S'\otimes\mathbb{R}}$ *an agent. Given a policy* $\pi : S \to \mathrm{D}A$*, samples in* $\Upsilon \cong S\otimes A\otimes\mathbb{R}\otimes S'$ *are given by Markov morphisms* $\mathcal{C}(I, \Upsilon\otimes M')$ *that result from the agent* $(S\otimes A, \mathrm{ev}, 1)$ *evaluating the policy* π *in the environment* $\mathbb{I}_{\mathrm{env}}$*.*

Proof. Let $\mathbb{I}_{\mathrm{env}}\left(\binom{S}{I}, \binom{A}{S'\otimes\mathbb{R}}\right)$ be an environment in a Markov category $\mathcal{C}$. Given the type of the environment $\mathbb{I}_{\mathrm{env}}$, for some $M, M' \in \mathcal{C}$ there exist $x_0 : \mathcal{C}(I, M\otimes S)$, $k : \mathcal{C}(M\otimes A, M'\otimes S'\otimes\mathbb{R})$ and $i : \mathcal{C}(M'\otimes I, M\otimes S)$. Likewise, for some $S : \mathcal{C}$ there exist $\mathrm{ev} : S\otimes(\mathrm{D}A)^S \to A\otimes S$. In the following derivation we label each homset with the representative element above.

$$
\begin{aligned}
&\mathcal{C}(I, \overset{x_0}{M}\otimes S)\times\mathcal{C}(I, \overset{\pi}{(\mathrm{D}A)^S})\times\mathcal{C}(S\otimes \overset{\mathrm{ev}}{(\mathrm{D}A)^S}, A)\times\mathcal{C}(M\otimes A, \overset{k}{M'}\otimes S'\otimes\mathbb{R})\\
&\to \mathcal{C}(I, M\otimes S\otimes(\mathrm{D}A)^S)\times\mathcal{C}(S\otimes(\mathrm{D}A)^S, A)\times\mathcal{C}(M\otimes A, M'\otimes S'\otimes\mathbb{R})\\
&\to \mathcal{C}(I, M\otimes S\otimes S\otimes(\mathrm{D}A)^S)\times\mathcal{C}(S\otimes(\mathrm{D}A)^S, A)\times\mathcal{C}(M\otimes A, M'\otimes S\otimes\mathbb{R})\\
&\to \mathcal{C}(I, M\otimes S\otimes A)\times\mathcal{C}(M\otimes A, M'\otimes S\otimes\mathbb{R})\\
&\to \mathcal{C}(I, S\otimes A\otimes M\otimes A)\times\mathcal{C}(M\otimes A, M'\otimes S\otimes\mathbb{R})\\
&\to \mathcal{C}(I, S\otimes A\otimes M'\otimes S'\otimes\mathbb{R})\\
&\cong \mathcal{C}(I, \Upsilon\otimes M')
\end{aligned}
$$

□

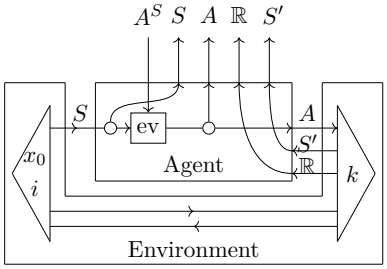
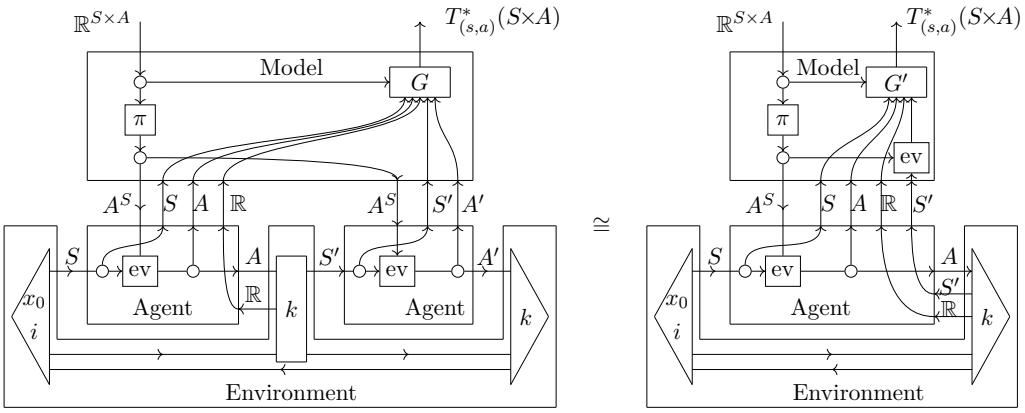
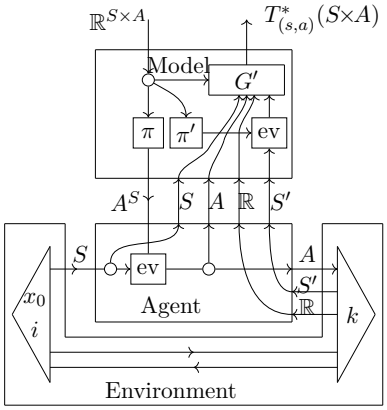


Figure 5.6: Composition of an agent and an environment for Q-learning.

Offline methods, unlike their online counterparts, interact with the agent only in a trivial way by showing samples to it. To reflect this, the residual types are given by $M = S \times A \times F$ 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 $\mathcal{C}(M' \otimes I, M \otimes S) \cong \mathcal{C}(I, M \otimes S)$ coincides with the initial state, which highlights the fact that the environment samples experiences (s, a, f) from a distribution defined by a dataset (figs. 5.8a and 5.8b).



(a) SARSA is on-policy.



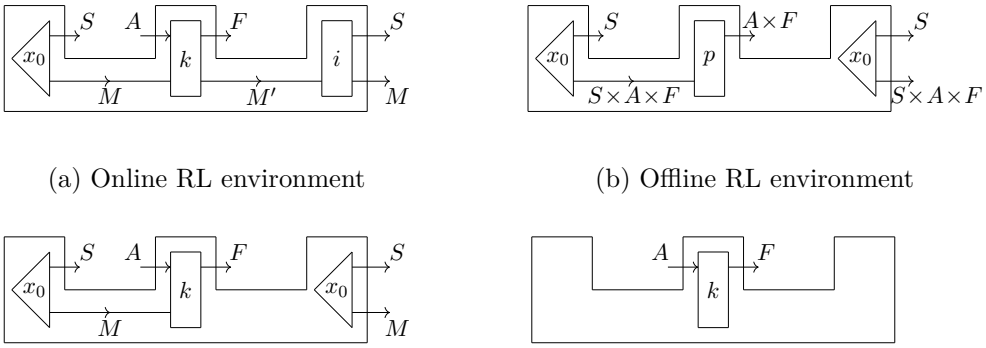
(b) Q-learning is off-policy.

Figure 5.7: Comparison between on-policy and off-policy algorithms.

To clarify the interplay between these the three structures described in this section, we look at the role played by internal and external policies in on- and off-policy methods. First, fig. 5.7a 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 diagram on the right.

On the other hand, Q-learning (fig. 5.7b) is an off-policy method, because the last action is computed by an internal policy $\pi' = \text{argmax}$ different from the one being deployed.



(a) Online RL environment

(b) Offline RL environment

(c) Contextual bandit environment

(d) Multi-armed bandit environment

Figure 5.8: Environments for RL and bandit problems as iteration contexts (definition 5.3.5). Omitted arrows are the unit. The offline continuation is a projection p of $A \times F$.

5.5.1 Prediction and bandit problems

The presented framework handles RL **prediction** problems for free in all the previous methods by trivialising the set $A = I$, 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 $\begin{pmatrix} \mathbb{R}^S \\ \Delta(\mathbb{R}^S) \end{pmatrix} \to \begin{pmatrix} I \\ S\mathbb{R}^n \end{pmatrix}$, and 1-TD prediction becomes $\begin{pmatrix} \mathbb{R}^S \\ \Delta(\mathbb{R}^S) \end{pmatrix} \to \begin{pmatrix} I \\ S\mathbb{R}S \end{pmatrix}$. 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 Markov reward process.

Moreover, **bandit problems** emerge by trivialising $M' = I$ (figs. 5.8c and 5.8d). 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 $k : A \to F$.

Chapter 6

Conclusion

This thesis has explored the rich interplay between Control Theory and Reinforcement Learning through the unifying lens of Category Theory. Our primary goals were firstly to illuminate structural commonalities and distinctions between these fields, particularly focusing on how their problem descriptions and solution methodologies can be expressed and analysed using categorical tools. Our hope is to offer a common language for interdisciplinary communication, potentially bridging the gap between control engineers, machine learning researchers, and category theorists. A secondary goal was to provide a constructive answer to an engineering problem: how is an optimal control actually constructed? This practical focus leads to an approach that may lack elegance from a category-theoretic perspective, highlighting a significant challenge encountered in interdisciplinary research.

Out of this effort, a core contribution of this work is the demonstration that fundamental constructs in both dynamic programming and optimal control, such as the Bellman operator and Riccati equations, can be captured as morphisms within categories of optics and decorated spans. In chapter 3, we showed how value improvement in Markov Decision Processes can be understood as optic composition, providing a novel categorical perspective on dynamic programming. Chapter 4 then focused on the more structured domain of Linear Quadratic Regulators, where we established the functoriality of Riccati difference equations for dynamics that are not necessarily time-invariant, and discussed how the control-estimation duality arises in this context. This duality,

realized as a functor between categories of biparameterised morphisms, highlights the deep structural equivalence between optimal control and state estimation problems. Finally, chapter 5 extended these ideas to the less structured realm of Reinforcement Learning, modelling empirical Bellman operators and RL agents and environments using parameterized optics, thereby providing an algebraic framework for understanding feedback and learning from samples.

To the applied control theorists, it lays groundwork on the design of domain specific languages for combinators for compositional MDP solutions. These would allow engineers to synthesize optimal control strategies for certain large-scale systems whose control flow follows the bidirectional processes composed via the data-accessor interpretation of lenses and the decorated span construction. Furthermore, the string diagrammatic syntax can provide a representation of these process that is not only illustrative but *formal*, thus aiding the implementation process in particular domains of knowledge.

To the category theory community, this thesis provides one (arguably opinionated) framework that clarifies some underlying mathematical structures of these applied fields. The categorical formalisms developed here, particularly the use of optics and decorated spans, open new avenues for generalization and the discovery of novel algorithms. For instance, the functorial treatment of Riccati equations suggests new ways to compose and transform optimal control solutions, while the optic-based models for RL provide a foundation for developing more compositionally sound and theoretically grounded learning algorithms. Ultimately, this work aims to equip researchers with a powerful toolkit to analyse, design, and innovate within the complex landscapes of control and learning.

6.1 Future work

Theorem 4.4.3 followed a linear algebra argument to show that Riccati equations produce optimal cost functions. For dynamic programming algorithms not over **FVec**, their convergence to optimality proof typically proceeds 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 Wasserstein metric between distributions. This would also be a reasonable place to unify our approach with the coalgebraic approach with metric coinduction [58], and the stream calculus on vector spaces [121].

Chapter 4 leaned heavily on objects of certain categories to be elements. It is difficult to talk about optimality of choices without comparison of elements of a space. This forces the definitions of functors like $\mathrm{Sym}_\bullet$ and $\mathrm{Sym}_{\bullet+}$ to be pointed. Luckily our choice of order between quadratic forms allowed collapsing linear maps and orders as a single class of morphisms in definition 4.2.9, but we believe that, in the same way decorated spans are a close construction to decorated cospans, a double categorical formalisation similar to open Markov processes [12] might lead to a more elegant formalisation of composition along dynamics. This also serves as a first step towards providing a unification with indexed optics with choice functions in the theory of open games, in particular to the solution concept of subgame-perfect equilibria.

Another point about chapter 4 is that not formalising processes with gaussian noise immediately as morphisms in e.g. the Markov category **Gauss** allowed us to define the duality between estimation algorithms and control algorithms as a functor, by making the isomorphic structure of covariances and quadratic costs explicit. The study of the location-scale family in section 4.2.3 and the fibrational perspective might be more appropriately formalised within the field of information geometry, where distributions are points of certain statistical manifolds. Many other dualities in control theory, as well as relations between hidden Markov models, Bayesian filters, etc. remain to be explored from this perspective and are part of future work. As an example, the control-estimation

duality functor $\mathbb{D}$ (definition 4.4.12) may be extended by an additional transformation of decorations $P \mapsto P^{-1}$ that assigns to every immediate cost matrix its inverse (which is still a symmetric positive semi-definite matrix), which defines the *precision matrix* (a Fisher information matrix for linear dynamics with gaussian noise [103]). Even though they have equivalent behaviour, there is significant computational trade-offs in the choice [6]. A construction analogous to theorem 4.4.13 may then realise a functorial version of the information filter and its information form duality [5, 124], and further avenues towards distributions other than gaussian might open also in the context of information geometry [3].

Continuous time MDPs pose a serious challenge to any approach for which categorical composition is sequential in time, as is done in this thesis, since composition of two morphisms in a category appears to be inherently discrete. (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 similarly arise as a fixpoint in this way. Exploring this systematically is important future work.

Appendix A

A.1 Miscellaneous

The concept of a monoidal category may be found in any textbook [115, Section VII.1][135, E.2] and is also covered in several reports and articles [155, 9]. The following definition is spelled out for completeness.

Definition A.1.1 (Monoidal category)**.** A category $\mathcal{C}$ is said to be a *monoidal category* if A *monoidal category* $\mathcal{C}$ is category with an additional structure of a monoidal product (a bifunctor) $\otimes\colon \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, and a monoidal unit $I \in \mathcal{C}_0$, which can both be seen as functors

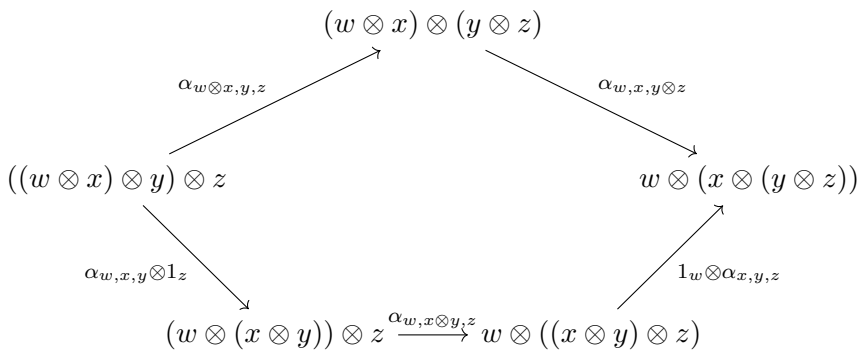
$$\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C} \qquad I : \star \to \mathcal{C}$$

Also, for given objects $a, b, c \in \mathcal{C}_0$, three natural isomorphisms:

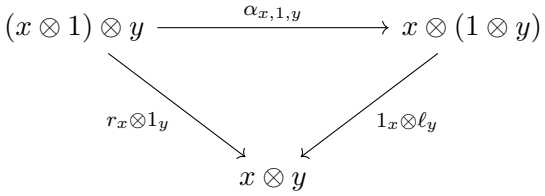
- the associator $\alpha_{a,b,c}\colon (a \otimes b) \otimes c \to a \otimes (b \otimes c)$
- the left unitor $\ell_a\colon I \otimes a \to a$
- the right unitor $r_a\colon a \otimes I \to a$

such that the following diagrams commute for all objects $w, x, y, z \in \mathcal{C}_0$:

- Associativity: the pentagon equation, governing the associator:



- Triangle equation, governing the unitors:



Definition A.1.2 (Delooping)**.** Given a monoidal category $(\mathcal{C}, \otimes, I)$, its *delooping* is the bicategory $\mathbb{B}\mathcal{C}$ which consists of a single 0-cell $*$ and a hom-category $\mathbb{B}\mathcal{C}(*,*)$ being $\mathcal{C}$.

Given a monoid M, we also call the category $\mathbb{B}M$ its delooping, responding to the fact that the delooping operation can be described for any (weak) k-tuply monoidal n-category and results in a $(k-1)$-tuply monoidal $(n+1)$-category [10].

Definition A.1.3 (Graph of a morphism)**.** In a cartesian category $\mathcal{C}$, the *graph* of a morphism $f : X \to Y$ is the image $\mathrm{gr}(f)$ of $X \xrightarrow{\Delta_X} X \times X \xrightarrow{X\times f} X \times Y$ that 'carves out' the product $X \times Y$. If $\mathcal{C}$ has pullbacks, the graph $\mathrm{gr}(f)$ is given by

$$\begin{array}{ccc} \mathrm{gr}(f) & \longrightarrow & Y \\ \downarrow & \lrcorner & \| \\ X & \xrightarrow[f]{} & Y \end{array}$$

which is canonically isomorphic to X itself. In $(\mathbf{FVec}, +, 0)$, the graph of $f : \mathbb{R}^n \to \mathbb{R}^m$ is the set

$$\mathrm{gr}(f) = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^n \oplus \mathbb{R}^m \mid f(\boldsymbol{x}) = \boldsymbol{y}\} \cong \mathbb{R}^n \subseteq \mathbb{R}^n \oplus \mathbb{R}^m$$

A.2 Linear algebra

Lemma A.2.1 (Sherman-Morrison-Woodbury)**.** *The* matrix inversion identity *is given by*

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1}+DA^{-1}B)^{-1}DA^{-1}$$

Definition A.2.2 (Schur complement)**.** For $p, q : \mathbb{N}$, let $M = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right)$ be a $(p+q)\times(p+q)$ block matrix. If D is invertible, the *Schur complement of D in M* is the $p \times p$ matrix defined by

$$M/D := A - BD^{-1}C$$

If A is invertible, the *Schur complement of A in M* is the $q \times q$ matrix defined by

$$M/A := D - CA^{-1}B$$

When D or A are singular, the *generalized Schur complement* is given by substituting a generalized inverse—like the Moore-Penrose pseudoinverse, axiomatized in the context of dagger categories [47]—for the inverses in M/D and M/A [36, A.5].

Lemma A.2.3. *Let $M = \left(\begin{smallmatrix} A & B \\ B^\top & D \end{smallmatrix}\right)$ and let M/D be the Schur complement of D in M. Then an expression of the inverse of M is the following:*

$$M^{-1} = \begin{bmatrix} I & 0 \\ -D^{-1}B^\top & I \end{bmatrix} \begin{bmatrix} (M/D)^{-1} & 0 \\ 0 & D^{-1} \end{bmatrix} \begin{bmatrix} I & -BD^{-1} \\ 0 & I \end{bmatrix}$$

Lemma A.2.4. *Let $M = \left(\begin{smallmatrix} A & B & C \\ B^\top & D & E \\ C^\top & E^\top & F \end{smallmatrix}\right)$ be a symmetric 3×3 block matrix. The Schur complement with respect to its lower right 2×2 block is given by*

$$M/\left(\begin{smallmatrix} D & E \\ E^\top & F \end{smallmatrix}\right) = A - (B - CF^{-1}E^\top)(D - EF^{-1}E^\top)^{-1}(B^\top - EF^{-1}C^\top) - CC^\top$$

Proof.

$$
\begin{aligned}
& M/\left(\begin{smallmatrix} D & E \\ E^\top & F \end{smallmatrix}\right) \\
=& A - \begin{bmatrix} B & C \end{bmatrix} \begin{bmatrix} D & E \\ E^\top & F \end{bmatrix}^{-1} \begin{bmatrix} B^\top \\ C^\top \end{bmatrix} && \text{(lemma A.2.3)} \\
=& A - \begin{bmatrix} B & C \end{bmatrix} \begin{bmatrix} I & 0 \\ -F^{-1}E^\top & I \end{bmatrix} \begin{bmatrix} (D - EF^{-1}E^\top)^{-1} & 0 \\ 0 & F^{-1} \end{bmatrix} \begin{bmatrix} I & -EF^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} B^\top \\ C^\top \end{bmatrix} \\
=& A - (B - CF^{-1}E^\top)(D - EF^{-1}E^\top)^{-1}(B^\top - EF^{-1}C^\top) - CF^{-1}C^\top
\end{aligned}
$$

□

Definition A.2.5. Let k be a field. A k-vector space A with an additional binary operation $\cdot : A \times A \to A$ is called a k-algebra if the following identities hold for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in A$ and all $a, b \in k$:

- Right distributivity: $(\boldsymbol{x} + \boldsymbol{y}) \cdot \boldsymbol{z} = \boldsymbol{x} \cdot \boldsymbol{z} + \boldsymbol{y} \cdot \boldsymbol{z}$
- Left distributivity: $\boldsymbol{z} \cdot (\boldsymbol{x} + \boldsymbol{y}) = \boldsymbol{z} \cdot \boldsymbol{x} + \boldsymbol{z} \cdot \boldsymbol{y}$
- Compatibility with scalars: $(a\boldsymbol{x}) \cdot (b\boldsymbol{y}) = (ab)(\boldsymbol{x} \cdot \boldsymbol{y})$

Definition A.2.6. The collection of all k-algebras forms a category $\mathbf{Alg}_k$ with morphisms being k-algebra homomorphisms, i.e. k-linear maps $k : A \to B$ that preserve the multiplication.

A.2.1 Algebraic Riccati equations in LQRs

The following proposition is a standard result in control theory; for completeness, we include it here with reference to standard treatments [154, 99]. Recall the statement of the linear quadratic regulator problem in section 4.4.

Proposition A.2.7. *Let $\boldsymbol{x}' = A\boldsymbol{x} + B\boldsymbol{u}$ be the state-space equation for a discrete linear dynamical system with $\boldsymbol{x}, \boldsymbol{x}' \in X \cong \mathbb{R}^n$ and $\boldsymbol{u} \in U \cong \mathbb{R}^m$ for some finite n, m. Let $\ell : X \oplus U \to \mathbb{R}$ be the symmetric bilinear function that defines the immediate cost of*

the system, given by a matrix $S \in \mathbb{R}^{n\times m}$ *and two positive definite matrices* $Q \in \mathbb{R}^{n\times n}$ *and* $R \in \mathbb{R}^{m\times m}$*:*

$$\ell(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{x}_k^\top Q \boldsymbol{x}_k + \boldsymbol{u}_k^\top R \boldsymbol{u}_k + \boldsymbol{x}_k^\top S \boldsymbol{u}_k + \boldsymbol{u}_k^\top S^\top \boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{x}_k^\top & \boldsymbol{u}_k^\top \end{bmatrix} \begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{bmatrix}$$

The optimal control $\boldsymbol{u}^* \in U$ *that minimizes the cumulative cost* $J(\boldsymbol{x}_0, \{\boldsymbol{u}_k\}_k) = \{\sum_k \ell(\boldsymbol{x}_k, \boldsymbol{u}_k) \mid \boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k\}$ *is given by a linear map of the state* $K : X \to U$*:*

$$\boldsymbol{u}_k^* = K\boldsymbol{x}_k := -(R + B^\top P_{k+1} B)^{-1}(S + B^\top P_{k+1} A)\boldsymbol{x}_k \tag{1}$$

Its associated optimal cost-to-go function $J^*(\boldsymbol{x}_0) = \boldsymbol{x}_0^\top P_0 \boldsymbol{x}_0$ *is given by the* Riccati difference equation *(RDE):*

$$P_k = Q + A^\top P_{k+1} A - (S^\top + A^\top P_{k+1} B)(R + B^\top P_{k+1} B)^{-1}(S + B^\top P_{k+1} A) \tag{2}$$

Proof. The total cost function can be expressed (co)recursively[1] as

$$J_k(\boldsymbol{x}_k, \{\boldsymbol{u}_i\}_{i=k}^T) = \ell(\boldsymbol{x}_k, \boldsymbol{u}_k) + J_{k+1}(A\boldsymbol{x}_k + B\boldsymbol{u}_k, \{u_i\}_{i=k+1}^T)$$

Moreover, the optimal total cost function J^* can also be written corecursively, as the minimization of quadratic forms is decomposable:

$$\begin{aligned} J_k^*(\boldsymbol{x}_k) &= \min_{\boldsymbol{u}_k, \ldots, \boldsymbol{u}_T} J_k(\boldsymbol{x}_k, \{\boldsymbol{u}_i\}_{i=k}^T) \\ &= \min_{\boldsymbol{u}_k} \left\{ \ell(\boldsymbol{x}_k, \boldsymbol{u}_k) + J_{k+1}^*(A\boldsymbol{x}_k + B\boldsymbol{u}_k^*) \right\} \end{aligned}$$

Assume $J_{k+1}^*(\boldsymbol{x}_{k+1}) = \boldsymbol{x}_{k+1}^\top P_{k+1} \boldsymbol{x}_{k+1}$ for some $P_{k+1} : \mathrm{Sym}_+^*(n)$. Because the cost

function is quadratic, it admits a unique minimum reached at the zero gradient point:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{u}_k} J_k(\boldsymbol{x}_k, \boldsymbol{u}_k) &= \frac{\partial}{\partial \boldsymbol{u}_k}\Big(\ell(\boldsymbol{x}_k, \boldsymbol{u}_k) + (A\boldsymbol{x}_k + B\boldsymbol{u}_k)^\top P_{k+1}(A\boldsymbol{x}_k + B\boldsymbol{u}_k)\Big) \\ &= 2\boldsymbol{u}_k^\top R + 2\boldsymbol{x}_k^\top S + 2(A\boldsymbol{x}_k + B\boldsymbol{u}_k)^\top P_{k+1} B \\ &\overset{!}{=} 0\end{aligned}$$

which yields the optimal control law (1). By substituting it into $J(\boldsymbol{x}_0, \{\boldsymbol{u}_k^*\}_k)$, we obtain equation (2). □

Remark A.2.8. When $(R + B^\top P_{k+1} B)$ is singular, we use the convention of substituting a generalized inverse for the inverses in (1) and (2).

Definition A.2.9. The *discrete algebraic Riccati equation* (DARE) is the fixpoint of the Riccati difference equation (2)

$$P = Q + A^\top P A - (S^\top + A^\top P B)(R + B^\top P B)^{-1}(S + B^\top P A)$$

whose solution P determines, in the context of infinite-horizon time-invariant LQRs, the steady state optimal cost.

Proposition A.2.10. *When the system cost does not have a state-action correlation component S, (2) simplifies to:*

$$P_k = A^\top P_{k+1}(I + G P_{k+1})^{-1} A + Q \tag{3}$$

with $G = B R^{-1} B^\top$.

Proof. Assign $S = 0$ to (2) and apply the matrix inversion lemma (lemma A.2.1). □

[1]The cost function is a *corecursive* function of the state-action pairs, as the cost of the current state-action pair is the sum of the immediate cost and the cost of the next state-action pair. However, its proof method is *recursive* as the trajectory horizon is assumed to be finite and takes as a base case the cost of the last state-action pair.

A.2.2 Kalman filter

The basic Kalman filter [100] is the optimal (minimum mean square, i.e. minimum variance) state estimator of a linear dynamical system with gaussian noise. Let the state-space equations be given by $\boldsymbol{x}' = A\boldsymbol{x} + \boldsymbol{w}$ and $\boldsymbol{y} = C\boldsymbol{x} + \boldsymbol{v}$, where $\boldsymbol{w}$ and $\boldsymbol{v}$ are zero-mean gaussian noises with variances $\mathbb{E}[\boldsymbol{w}\boldsymbol{w}^\top] = W$ and $\mathbb{E}[\boldsymbol{v}\boldsymbol{v}^\top] = V$. Note that the probability distribution function f of a zero-mean Gaussian random variable $\boldsymbol{w} \sim \mathcal{N}(0, W)$ has the form

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n \det(W)}} \exp\left(-\frac{1}{2}x^\top W^{-1} x\right)$$

and thus the distribution is characterized by the symmetric quadratic form W.

Estimated values are denoted with a hat $\hat{(\,)}$, and can be either *priors* or *posteriors*. *Priors* are given by predictions from past steps; for example, $\hat{\boldsymbol{x}}_{k+1|k}$ is the prior on the state at step $k+1$ predicted from information available at step k. *Posteriors* are corrections on priors given by new measurements or observations. The posterior state given information available at step $k+1$ is denoted $\hat{\boldsymbol{x}}_{k+1|k+1}$.

Briefly, the Kalman filter assumes an estimation of the initial state $\hat{\boldsymbol{x}}_{0|0}$ and an uncertainty Σ, and applies iteratively the following operations for each step k:

1. Predict the next state $\hat{\boldsymbol{x}}_{k+1|k}$ and error[2] $\Sigma_{k+1|k}$:

$$\begin{aligned}
\hat{\boldsymbol{x}}_{k+1|k} &= A\hat{\boldsymbol{x}}_{k|k} \\
\Sigma_{k+1|k} &= \mathbb{E}[(\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1|k})(\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1|k})^\top] \\
&= \mathbb{E}[(A\boldsymbol{x}_k + \boldsymbol{w}_k - A\hat{\boldsymbol{x}}_{k|k})(A\boldsymbol{x}_k + \boldsymbol{w}_k - A\hat{\boldsymbol{x}}_{k|k})^\top] \\
&= A\mathbb{E}[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k})(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k})^\top]A^\top + \mathbb{E}[\boldsymbol{w}_k \boldsymbol{w}_k^\top] \\
&= A\Sigma_{k|k}A^\top + W
\end{aligned} \tag{4}$$

2. Measure $\boldsymbol{y}_{k+1}$, calculate the *innovation* or *residual* $\boldsymbol{z}_{k+1}$ and innovation covariance

[2] The second to last step of the error prediction is why we need the model to be linear.

error ζ_{k+1}:

$$\begin{aligned}
\boldsymbol{z}_{k+1} &= \boldsymbol{y}_{k+1} - C\hat{\boldsymbol{x}}_{k+1|k} \\
&= C\boldsymbol{x}_{k+1} + \boldsymbol{v}_{k+1} - C\hat{\boldsymbol{x}}_{k+1|k} \\
\zeta_{k+1} &= \mathbb{E}[\boldsymbol{z}_{k+1}\boldsymbol{z}_{k+1}^\top] \\
&= \mathbb{E}[(C\boldsymbol{x}_{k+1} + \boldsymbol{v}_{k+1} - C\hat{\boldsymbol{x}}_{k+1|k})(C\boldsymbol{x}_{k+1} + \boldsymbol{v}_{k+1} - C\hat{\boldsymbol{x}}_{k+1|k})^\top] \\
&= C\mathbb{E}[(\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1|k})(\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1|k})^\top]C^\top + \mathbb{E}[\boldsymbol{v}_{k+1}\boldsymbol{v}_{k+1}^\top] \\
&= C\Sigma_{k+1|k}C^\top + V
\end{aligned}$$

3. Calculate the optimal gain (*Kalman gain*) K_{k+1}^*: The minimization of the trace of the *estimation covariance* $\Sigma_{k+1|k+1}$ is given by the first-order optimality condition

$$\partial_{K_{k+1}} \operatorname{Tr}(\Sigma_{k+1|k+1})\Big|_{K_{k+1}=K_{k+1}^*} = 0$$

which results in

$$K_{k+1}^* = \Sigma_{k+1|k}C^\top(C\Sigma_{k+1|k}C^\top + V)^{-1} \tag{5}$$

4. Update the prediction of state and error to the posteriors $\hat{\boldsymbol{x}}_{k+1|k+1}$ and $\Sigma_{k+1|k+1}$:

$$\begin{aligned}
\hat{\boldsymbol{x}}_{k+1|k+1} &= \hat{\boldsymbol{x}}_{k+1|k} + K_{k+1}\boldsymbol{z}_{k+1} \\
\Sigma_{k+1|k+1} &= \mathbb{E}[(\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1|k+1})(\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1|k+1})^\top] \\
&= \cdots \\
&= (I - K_{k+1}C)\Sigma_{k+1|k}(I - K_{k+1}C)^\top + K_{k+1}VK_{k+1}^\top
\end{aligned}$$

This last expressions is known as the Joseph form [38]. After substituting (4) and the optimal gain (5) and a bit of algebra, we get the expression of the optimal posterior

covariance $\Sigma_{k+1|k+1}$ in terms of the previous posterior $\Sigma_{k|k}$ for a gain K_{k+1}:

$$\Sigma^*_{k+1|k+1} = A\Sigma_{k|k}A^\top + W - (A\Sigma_{k|k}A^\top + W)C^\top (C(A\Sigma_{k|k}A^\top + W)C^\top + V)^{-1} C(A\Sigma_{k|k}A^\top + W) \tag{6}$$

Bibliography

[1] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. "Categories of containers". In: *Proceedings of the 6th International Conference on Foundations of Software Science and Computation Structures and Joint European Conference on Theory and Practice of Software*. FOSSACS'03/ETAPS'03. Warsaw, Poland, 2003, pp. 23–38. ISBN: 3540008977.

[2] Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-agent reinforcement learning. Foundations and modern approaches*. Cambridge: The MIT Press, 2024. 368 pp. ISBN: 9780262049375.

[3] Shun-ichi Amari. *Information Geometry and Its Applications*. Springer Japan, 2016. ISBN: 9784431559788. DOI: 10.1007/978-4-431-55978-8.

[4] Brian D. Anderson and John Barratt Moore. *Optimal control: Linear Quadratic Methods. Linear quadratic methods*. Prentice Hall information and system sciences series. Englewood Cliffs, N.J.: Prentice Hall, 1989. 380 pp. ISBN: 0136386512.

[5] Brian D. O. Anderson. *Optimal filtering*. Ed. by John B. Moore. Dover ed. Dover books on engineering. Mineola, N.Y.: Dover Publications, 2005. 1357 pp. ISBN: 1621986047.

[6] Nicholas Assimakis, Maria Adam, and Anargyros Douladiris. "Information Filter and Kalman Filter Comparison: Selection of the Faster Filter". In: *International Journal of Information Engineering (IJIE)* 2 (Jan. 2012), pp. 1–5.

[7] K.J Åström. "Optimal control of Markov processes with incomplete state information". In: *Journal of Mathematical Analysis and Applications* 10.1 (Feb. 1965), pp. 174–205. DOI: 10.1016/0022-247x(65)90154-x.

[8] K.J. Åström and K. Furuta. "Swinging up a pendulum by energy control". In: *Automatica* 36.2 (Feb. 2000), pp. 287–295. ISSN: 0005-1098. DOI: 10.1016/s0005-1098(99)00140-5.

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

[10] John C. Baez and James Dolan. *Categorification*. 1998. DOI: 10.1090/conm/230/03336.

[11] John C. Baez and Brendan Fong. "A Compositional Framework for Passive Linear Networks". In: *Theory and Applications of Categories* 33 (2015), pp. 1158–1222. arXiv: 1504.05625 [math.CT].

[12] John C. Baez, Brendan Fong, and Blake S. Pollard. "A compositional framework for Markov processes". In: *Journal of Mathematical Physics* 57.3 (Mar. 2016). ISSN: 1089-7658. DOI: 10.1063/1.4941578.

[13] John C. Baez and Jade Master. "Open Petri nets". In: *Mathematical Structures in Computer Science* 30.3 (Mar. 2020), pp. 314–341. ISSN: 1469-8072. DOI: 10.1017/s0960129520000043.

[14] Georgios Bakirtzis, Michail Savvas, and Ufuk Topcu. *Categorical semantics of compositional reinforcement learning*. 2022. arXiv: 2208.13687 [cs.AI].

[15] Wolfram Barfuss. "Learning dynamics and decision paradigms in social-ecological dilemmas". en. PhD thesis. Humboldt-Universität zu Berlin, 2019. DOI: 10.18452/20127.

[16] E. Barnard. "Temporal-difference methods and Markov models". In: *IEEE Transactions on Systems, Man, and Cybernetics* 23.2 (1993), pp. 357–365. ISSN: 0018-9472. DOI: 10.1109/21.229449.

[17] Richard Bellman. "Dynamic Programming". English. In: *Princeton University Press* (Dec. 1957). DOI: 10.1515/9781400835386.

[18] Jean Bénabou. "Introduction to bicategories". In: *Reports of the Midwest Category Seminar*. Springer Berlin Heidelberg, 1967, pp. 1–77. ISBN: 9783540355458. DOI: 10.1007/bfb0074299.

[19] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer New York, 1985. ISBN: 9781475742862. DOI: 10.1007/978-1-4757-4286-2.

[20] Mattia G. Bergomi and Pietro Vertechi. *Neural network layers as parametric spans*. 2022. arXiv: 2208.00809 [math.CT].

[21] Dimitri P. Bertsekas. "Distributed dynamic programming". In: *IEEE Transactions on Automatic Control* 27.3 (June 1982), pp. 610–616. ISSN: 0018-9286. DOI: 10.1109/tac.1982.1102980.

[22] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Third. Athena Scientific, 2007, p. 464. ISBN: 9781886529304. DOI: https://dl.acm.org/doi/10.5555/1396348.

[23] Dimitri P. Bertsekas. *Neuro-dynamic programming*. Ed. by John N. Tsitsiklis. 3rd printing. Literaturverzeichnis: S. 475-486. Belmont, Massachusetts: Athena Scientific, 2016. 491 pp. ISBN: 9781886529106.

[24] Dimitri P. Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific optimization and computation series. Athena Scientific, 2019. ISBN: 9781886529397.

[25] Rajendra Bhatia. *Matrix Analysis*. Springer New York, 1997. ISBN: 9781461206538. DOI: 10.1007/978-1-4612-0653-8.

[26] Christopher M. Bishop. *Pattern recognition and machine learning*. Information Science and Statistics. New York, NY: Springer Science+Business Media, LLC, 2006. 758 pp.

[27] Joe Bolt, Jules Hedges, and Philipp Zahn. "Bayesian open games". In: *Compositionality* 5.9 (2023).

[28] Filippo Bonchi, Alessandro Di Giorgio, and Fabio Zanasi. “From Farkas’ Lemma to Linear Programming: an Exercise in Diagrammatic Algebra”. en. In: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. DOI: 10.4230/LIPICS.CALCO.2021.9.

[29] Filippo Bonchi, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. “Graphical Affine Algebra”. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, June 2019, pp. 1–12. DOI: 10.1109/lics.2019.8785877.

[30] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. “A Categorical Semantics of Signal Flow Graphs”. In: *CONCUR 2014 – Concurrency Theory*. Springer Berlin Heidelberg, 2014, pp. 435–450. ISBN: 9783662445846. DOI: 10.1007/978-3-662-44584-6_30.

[31] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. “Full Abstraction for Signal Flow Graphs”. In: *ACM SIGPLAN Notices* 50.1 (Jan. 2015), pp. 515–526. ISSN: 1558-1160. DOI: 10.1145/2775051.2676993.

[32] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. “Interacting Hopf algebras”. In: *Journal of Pure and Applied Algebra* 221.1 (Jan. 2017), pp. 144–184. ISSN: 0022-4049. DOI: 10.1016/j.jpaa.2016.06.002.

[33] Ugo Boscain and Mario Sigalotti. “Introduction to Controllability of Nonlinear Systems”. In: *Contemporary Research in Elliptic PDEs and Related Topics*. Springer International Publishing, 2019, pp. 203–219. ISBN: 9783030189211. DOI: 10.1007/978-3-030-18921-1_4.

[34] Nicola Botta, Patrik Jansson, Cezar Ionescu, David R. Christiansen, and Edwin Brady. “Sequential decision problems, dependent types and generic solutions”. In: *Logical Methods in Computer Science* 13.1 (Mar. 2017). ISSN: 1860-5974. DOI: 10.23638/lmcs-13(1:7)2017.

[35] Leon Bottou and Olivier Bousquet. “The tradeoffs of large scale learning”. In: *Proceedings of the 21st International Conference on Neural Information Process-*

ing Systems. NIPS'07. Vancouver, British Columbia, Canada: Curran Associates Inc., 2007, pp. 161–168. ISBN: 9781605603520. DOI: 10.5555/2981562.2981583.

[36] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Mar. 2004. ISBN: 9780511804441. DOI: 10.1017/cbo9780511804441.

[37] Dylan Braithwaite, Jules Hedges, and Toby St Clere Smithe. "The Compositional Structure of Bayesian Inference". en. In: *Proceedings of Mathematical Foundations of Computer Science 2023*. Vol. 272. Leibniz Proceedings in Informatics 24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/LIPICS.MFCS.2023.24.

[38] Richard S. Bucy. *Filtering for Stochastic Processes with Applications to Guidance*. Ed. by Peter D. Joseph. AMS Chelsea Publishing Ser. v.326. Providence: American Mathematical Society, 2005. 1238 pp. ISBN: 9781470467715.

[39] Matteo Cappuci and David Jaz Myers. "Constructing triple categories of cybernetics processes". 2023.

[40] Matteo Capucci. "Diegetic Representation of Feedback in Open Games". In: *Electronic Proceedings in Theoretical Computer Science* 380 (Aug. 2023), pp. 145–158. ISSN: 2075-2180. DOI: 10.4204/eptcs.380.9.

[41] Matteo Capucci and Bruno Gavranović. *Actegories for the working amthematician*. 2022. arXiv: 2203.16351 [math.CT].

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

[43] Matteo Capucci, Neil Ghani, Jérémy Ledent, and Fredrik Nordvall Forsberg. "Translating extensive form games to open games with agency". In: *Electronic Proceedings in Theoretical Computer Science* 372 (Nov. 2022), pp. 221–234. DOI: 10.4204/eptcs.372.16.

[44] George Casella and Roger L. Berger. *Statistical inference*. Second. Pacific Grove, Calif.: Duxbury, 2002. 660 pp. ISBN: 0534243126.

[45] Kenta Cho and Bart Jacobs. "Disintegration and Bayesian inversion via string diagrams". In: *Mathematical Structures in Computer Science* 29.7 (Mar. 2019), pp. 938–971. ISSN: 1469-8072. DOI: 10.1017/s0960129518000488.

[46] Bryce Clarke, Derek Elkins, Jeremy Gibbons, Fosco Loregian, Bartosz Milewski, Emily Pillmore, and Mario Román. "Profunctor Optics, a Categorical Update". In: *Compositionality* 6.1 (Feb. 2024), p. 1. ISSN: 2631-4444. DOI: 10.32408/compositionality-6-1.

[47] Robin Cockett and Jean-Simon Pacaud Lemay. "Moore-Penrose Dagger Categories". In: *Electronic Proceedings in Theoretical Computer Science* 384 (Aug. 2023), pp. 171–186. ISSN: 2075-2180. DOI: 10.4204/eptcs.384.10.

[48] Geoffrey Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson, and Fabio Zanasi. "Categorical foundations of gradient-based learning". In: *Proceedings of ESOP 2022*. Vol. 13240. Lecture Notes in Computer Science. 2022.

[49] Davidad Dalrymple. "Dioptics: a common generalization of open games and gradient-based learners". Unpublished. 2019. URL: https://research.protocol.ai/publications/dioptics-a-common-generalization-of-open-games-and-gradient-based-learners/dalrymple2019.pdf.

[50] Robert Dawson, Robert Paré, and Dorette Pronk. "The span construction". In: *Theory and Applications of Categories* 24.13 (2010), pp. 302–377.

[51] Pierre Del Moral and Emma Horton. "A Note on Riccati Matrix Difference Equations". In: *SIAM Journal on Control and Optimization* 60.3 (May 2022), pp. 1393–1409. ISSN: 1095-7138. DOI: 10.1137/21m1437226.

[52] Antonin Delpeuch. "Autonomization of Monoidal Categories". In: *Proceedings of Applied Category Theory 2019*. Vol. 323. Open Publishing Association, Sept. 2020, pp. 24–43. DOI: 10.4204/eptcs.323.3.

[53] Eric V. Denardo. "Contraction Mappings in the Theory Underlying Dynamic Programming". In: *SIAM Review* 9.2 (Apr. 1967), pp. 165–177. DOI: 10.1137/1009030.

[54] Zinovy Diskin, Yingfei Xiong, and Krzysztof Czarnecki. "From State- to Delta-Based Bidirectional Model Transformations: the Asymmetric Case." In: *The Journal of Object Technology* 10 (2011), 6:1. ISSN: 1660-1769. DOI: 10.5381/jot.2011.10.1.a6.

[55] Jason Michael Erbele. "Categories in control: applied PROPs". PhD thesis. University of California Riverside, 2016. arXiv: 1611.07591 [math.CT].

[56] Nicolas Eschenbaum, Filip Mellgren, and Philipp Zahn. *Robust algorithmic collusion*. 2022. arXiv: 2201.00345 [econ.GN].

[57] Norman Ferns, Prakash Panangaden, and Doina Precup. *Metrics for Finite Markov Decision Processes*. 2012. DOI: doi/10.5555/1036843.1036863.

[58] Frank M. V. Feys, Helle Hvid Hansen, and Lawrence S. Moss. "Long-Term Values in Markov Decision Processes, (Co)Algebraically". In: *Coalgebraic Methods in Computer Science*. Ed. by Corina Cîrstea. Vol. 11202. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 78–99. ISBN: 978-3-030-00389-0. DOI: 10.1007/978-3-030-00389-0_6.

[59] Brendan Fong. "The Algebra of Open and Interconnected Systems". PhD thesis. University of Oxford, 2016. arXiv: 1609.05382 [math.CT].

[60] Brendan Fong and Maru Sarazola. *A recipe for black box functors*. 2018. arXiv: 1812.03601 [math.CT].

[61] Brendan Fong, Paweł Sobociński, and Paolo Rapisarda. "A categorical approach to open and interconnected dynamical systems". In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '16. ACM, July 2016, pp. 495–504. DOI: 10.1145/2933575.2934556.

[62] Brendan Fong and David I Spivak. *Hypergraph Categories*. 2018. arXiv: 1806.08304 [math.CT].

[63] Brendan Fong and David I Spivak. *Seven Sketches in Compositionality: An Invitation to Applied Category Theory*. 2018. arXiv: 1803.05316 [math.CT].

[64] J. Nathan Foster, Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierce, and Alan Schmitt. "Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem". In: *ACM Transactions on Programming Languages and Systems* 29.3 (May 2007), p. 17. ISSN: 1558-4593. DOI: 10.1145/1232420.1232424.

[65] Luisa de Francesco Albasini, Nicoletta Sabadini, and Robert F. C. Walters. "The Compositional Construction of Markov Processes". In: *Applied Categorical Structures* 19.1 (July 2010), pp. 425–437. ISSN: 1572-9095. DOI: 10.1007/s10485-010-9233-0.

[66] Bernard Friedland. *Control Systems Design: An Introduction to State-Space Methods. An Introduction To State-Space Methods*. McGraw-Hill Companies, 1985, p. 513. ISBN: 9780070224414.

[67] Tobias Fritz. "A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics". en. In: *Advances in Mathematics* 370 (Aug. 2020), p. 107239. ISSN: 00018708. DOI: 10.1016/j.aim.2020.107239.

[68] Tobias Fritz. *Convex spaces I: Definitions and examples*. 2009. arXiv: 0903.5522 [math.MG].

[69] Bruno Gavranović. "Fundamental components of deep learning: A category-theoretic approach". PhD thesis. University of Strathclyde, 2024. arXiv: 2403.13001 [cs.LG].

[70] Bruno Gavranović. *Space-time tradeoffs of lenses and optics via higher category theory*. 2022. arXiv: 2209.09351 [math.CT].

[71] Neil Ghani, Jules Hedges, Viktor Winschel, and Philipp Zahn. "Compositional game theory". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. Oxford, United Kingdom: Association for Computing Machinery (ACM), 2018, pp. 472–481. ISBN: 9781450355834. DOI: 10.1145/3209108.3209165.

[72] Robert Ghrist. "Barcodes: The persistent topology of data". In: *Bulletin of the American Mathematical Society* 45.01 (Oct. 2007), pp. 61–76. ISSN: 0273-0979. DOI: 10.1090/s0273-0979-07-01191-3.

[73] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016. ISBN: 9780262035613.

[74] Igor Griva, Stephen G. Nash, and Ariela Sofer. *Linear and Nonlinear Optimization*. SIAM, Jan. 2009. ISBN: 9780898716610. DOI: 10.1137/1.9780898717730.

[75] Alexander Grothendieck. "Technique de descente et théorèmes d'existence en géométrie algébrique. I. Généralités. Descente par morphismes fidèlement plats". fr. In: *Séminaire Bourbaki : années 1958/59 - 1959/60, exposés 169-204*. Astérisque 5. talk:190. Société mathématique de France, 1960, pp. 299–327. URL: https://www.numdam.org/item/SB_1958-1960__5__299_0/.

[76] Tyler Hanks, Baike She, Matthew Hale, Evan Patterson, Matthew Klawonn, and James Fairbanks. *Modeling Model Predictive Control: A Category Theoretic Framework for Multistage Control Problems*. 2023. arXiv: 2305.03820 [math.OC].

[77] William B. Haskell, Rahul Jain, and Dileep Kalathil. *Empirical Dynamic Programming*. 2013. arXiv: 1311.5918 [math.OC].

[78] Jules Hedges. *Iteration with optics*. Blog post. 2024. URL: https://cybercat.institute/2024/02/22/iteration-optics/.

[79] Jules Hedges. *Lenses for philosophers*. Blog post. 2018. URL: https://julesh.com/2018/08/16/lenses-for-philosophers/.

[80] Jules Hedges. "Morphisms of open games". In: *Proceedings of MFPS 2018*. Vol. 341. Elsevier BV, Dec. 2018, pp. 151–177. DOI: 10.1016/j.entcs.2018.11.008.

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

[82] Jules Hedges. "Towards compositional game theory". PhD thesis. Queen Mary University of London, 2016.

[83] Jules Hedges and Wolfram Barfuss. *EcologicalPublicGood.hs – part of the Open Game Engine*. Accessed: 2024-02-10. 2019. URL: https://github.com/jules-hedges/open-games-hs/blob/og-v0.1/src/OpenGames/Examples/EcologicalPublicGood/EcologicalPublicGood.hs.

[84] Jules Hedges and Riu Rodríguez Sakamoto. "Reinforcement Learning in Categorical Cybernetics". In: (Apr. 2024). arXiv: 2404.02688 [cs.LG].

[85] Jules Hedges and Riu Rodríguez Sakamoto. "Value Iteration is Optic Composition". In: *Proceedings of Applied Category Theory 2022*. Vol. 380. Open Publishing Association, Aug. 2023, pp. 417–432. DOI: 10.4204/eptcs.380.24.

[86] James Hefford and Cole Comfort. "Coend Optics for Quantum Combs". In: *Electronic Proceedings in Theoretical Computer Science* 380 (Aug. 2023), pp. 63–76. ISSN: 2075-2180. DOI: 10.4204/eptcs.380.4.

[87] Claudio Hermida. "Representable Multicategories". In: *Advances in Mathematics* 151.2 (May 2000), pp. 164–225. ISSN: 0001-8708. DOI: 10.1006/aima.1999.1877.

[88] Claudio Hermida and Robert D Tennent. "Monoidal indeterminates and categories of possible worlds". In: *Theoretical Computer Science* 430 (Apr. 2012), pp. 3–22. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2012.01.001.

[89] Chris Heunen. "Semimodule Enrichment". In: *Electronic Notes in Theoretical Computer Science* 218 (Oct. 2008), pp. 193–208. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2008.10.012.

[90] Chris Heunen and Jamie Vicary. *Categories for Quantum Theory: An Introduction*. Oxford University Press, Nov. 2019. ISBN: 9780191802584. DOI: 10.1093/oso/9780198739623.001.0001.

[91] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 1530-888X. DOI: 10.1162/neco.1997.9.8.1735.

[92] Bart Jacobs. *Categorical logic and type theory*. paperback ed. Studies in logic and the foundations of mathematics 141. Amsterdam [u.a.]: Elsevier, 2001. 760 pp. ISBN: 9780444508539.

[93] George Janelidze and Gregory M Kelly. "A note on actions of a monoidal category". In: *Theory and Applications of Categories* 9.61-91 (2001), p. 02.

[94] Andrew H. Jazwinski. *Stochastic processes and filtering theory*. 12. Mathematics in science and engineering 64. Literaturverz. S. 365 - 366. San Diego [u.a.]: Acad. Press, 1997. 376 pp. ISBN: 0123815509.

[95] Michael Johnson and Robert Rosebrugh. "Delta Lenses and Opfibrations". en. In: (2013). DOI: 10.14279/TUJ.ECEASST.57.875.

[96] Peter T. Johnstone, Robert Paré, R. D. Rosebrugh, D. Schumacher, R. J. Wood, and G. C. Wraith. *Indexed Categories and Their Applications*. Springer Berlin Heidelberg, 1978. ISBN: 9783540357629. DOI: 10.1007/bfb0061360.

[97] André Joyal and Ross Street. "The geometry of tensor calculus, I". In: *Advances in Mathematics* 88.1 (July 1991), pp. 55–112. ISSN: 0001-8708. DOI: 10.1016/0001-8708(91)90003-p.

[98] L. P. Kaelbling, M. L. Littman, and A. W. Moore. "Reinforcement Learning: A Survey". In: *Journal of Artificial Intelligence Research* 4 (May 1996), pp. 237–285. ISSN: 1076-9757. DOI: 10.1613/jair.301.

[99] Thomas Kailath. *Linear systems*. Prentice-Hall information and system sciences series. Literaturangaben. Englewood Cliffs, NJ: Prentice-Hall, 1980. 682 pp. ISBN: 0135369614.

[100] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552.

[101] Piergiulio Katis, N. Sabadini, and Robert F. C. Walters. "Span(Graph): A categorical algebra of transition systems". In: *Algebraic Methodology and Software Technology*. Springer Berlin Heidelberg, 1997, pp. 307–321. ISBN: 9783540696612. DOI: 10.1007/bfb0000479.

[102] Anatole Katok and Boris Hasselblatt. *Introduction to the modern theory of dynamical systems*. Eigth. Encyclopedia of mathematics and its applications 54. Cambridge Univ. Press, 2006. 802 pp. ISBN: 9780521341875.

[103] Robert W. Keener. *Theoretical Statistics: Topics for a Core Course*. Springer New York, 2010. ISBN: 9780387938394. DOI: 10.1007/978-0-387-93839-4.

[104] G.M. Kelly. *Basic concepts of enriched category theory*. Vol. 64. Lecture Notes in Mathematics. Cambridge University Press, 1982.

[105] Aleks Kissinger. *Finite matrices are complete for (dagger-)hypergraph categories*. 2014. arXiv: 1406.5942 [math.CT].

[106] Anders Kock. "Closed categories generated by commutative monads". In: *Journal of the Australian Mathematical Society* 12.4 (Nov. 1971), pp. 405–424. DOI: 10.1017/s1446788700010272.

[107] Vijay Konda and John Tsitsiklis. "Actor-Critic Algorithms". In: *Society for Industrial and Applied Mathematics* 42 (Apr. 2001).

[108] Dexter Kozen and Nicholas Ruozzi. "Applications of Metric Coinduction". In: *Logical Methods in Computer Science* Volume 5, Issue 3 (Sept. 2009). ISSN: 1860-5974. DOI: 10.2168/lmcs-5(3:10)2009.

[109] Stephen Lack. "Composing PROPs". In: *Theory and Applications of Categories* 13.9 (2004), pp. 147–163. ISSN: 1201-561X.

[110] Serge Lang. *Fundamentals of Differential Geometry*. Springer New York, 1999. ISBN: 9781461205418. DOI: 10.1007/978-1-4612-0541-8.

[111] Elena di Lavore, Giovanni de Felice, and Mario Román. "Monoidal Streams for Dataflow Programming". In: *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '22. New York, NY, USA: Association for Computing Machinery (ACM), 2022. ISBN: 9781450393515. DOI: 10.1145/3531130.3533365.

[112] Tom Leinster. *Basic Category Theory*. Cambridge University Press, July 2014. ISBN: 9781107360068. DOI: 10.1017/cbo9781107360068.

[113] Fosco Loregian. *(Co)end calculus*. Cambridge University Press, June 2021. ISBN: 9781108746120. DOI: 10.1017/9781108778657. arXiv: 1501.02503 [math.CT].

[114] Karl Löwner. "Über monotone Matrixfunktionen". In: *Mathematische Zeitschrift* 38.1 (Dec. 1934), pp. 177–216. ISSN: 1432-1823. DOI: 10.1007/bf01170633.

[115] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer New York, 1978. ISBN: 9781475747218. DOI: 10.1007/978-1-4757-4721-8.

[116] Saunders MacLane. "Categorical algebra". In: *Bulletin of the American Mathematical Society* 71.1 (1965), pp. 40–106.

[117] P. Marbach and J.N. Tsitsiklis. "Simulation-based optimization of Markov reward processes". In: *IEEE Transactions on Automatic Control* 46.2 (2001), pp. 191–209. ISSN: 0018-9286. DOI: 10.1109/9.905687.

[118] Albert W. Marshall, Ingram Olkin, and Barry C. Arnold. *Inequalities: Theory of Majorization and Its Applications*. Springer New York, 2011. ISBN: 9780387682761. DOI: 10.1007/978-0-387-68276-1.

[119] Peter S. Maybeck. *Stochastic models, estimation and control*. Mathematics in science and engineering v. 141. New York: Academic Press, 1979. 423 pp. ISBN: 9780080956503.

[120] Sean Meyn. *Control Systems and Reinforcement Learning*. Cambridge University Press, May 2022. ISBN: 9781316511961. DOI: 10.1017/9781009051873.

[121] Stefan Milius. "A Sound and Complete Calculus for Finite Stream Circuits". In: *2010 25th Annual IEEE Symposium on Logic in Computer Science*. IEEE, July 2010. DOI: 10.1109/lics.2010.11.

[122] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG].

[123] Joe Moeller and Christina Vasilakopoulou. "Monoidal Grothendieck construction". In: *Theory and Applications of Categories* 35.31 (2020), pp. 1159–1207. arXiv: 1809.00727 [math.CT].

[124] Arthur G.O. Mutambara. *Decentralized Estimation and Control for Multisensor Systems*. May 2019. DOI: 10.1201/9781315140803.

[125] David Jaz Myers. "Categorical systems theory". Draft book. 2023. URL: https://www.davidjaz.com/Papers/DynamicalBook.pdf.

[126] David Jaz Myers. "Double Categories of Open Dynamical Systems (Extended Abstract)". en. In: *Electronic Proceedings in Theoretical Computer Science* 333 (Feb. 2021), pp. 154–167. ISSN: 2075-2180. DOI: 10.4204/eptcs.333.11.

[127] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton, NJ [u.a.]: Princeton University Press, 1953. 739 pp. ISBN: 9781400829460.

[128] Nelson Niu and David I. Spivak. *Polynomial Functors: A Mathematical Theory of Interaction*. Cambridge University Press, Sept. 2025. ISBN: 9781009576710. DOI: 10.1017/9781009576734.

[129] nLab authors. *stuff, structure, property*. Revision 58. Mar. 2025. URL: https://ncatlab.org/nlab/show/stuff%2C+structure%2C+property.

[130] Jorge Nocedal. *Numerical optimization*. Ed. by Stephen J. Wright. Second. Springer series in operation research and financial engineering. New York, NY: Springer, 2006. 664 pp. ISBN: 1493937111.

[131] Craig Pastro and Ross Street. "Doubles for monoidal categories". In: *Theory and Applications of Categories* 21.4 (2008).

[132] Matthew Pickering, Jeremy Gibbons, and Nicolas Wu. "Profunctor Optics: Modular Data Accessors". In: *The Art, Science, and Engineering of Programming* 1.2 (Apr. 2017). ISSN: 2473-7321. DOI: 10.22152/programming-journal.org/2017/1/7.

[133] Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. en. Wiley series in probability and statistics. Hoboken, NJ: Wiley-Interscience, 2005. ISBN: 978-0-471-72782-8.

[134] Jacopo Riccati. *Animadversiones in aequationes differentiales secundi gradus*. Actorum Eruditorum quae Lipsiae publicantur Supplementa 8. prostant apud Joh. Grossii haeredes & J.F. Gleditschium, 1724, pp. 66–73.

[135] Emily Riehl. *Category theory in context*. Aurora: Dover modern math originals. Mineola, New York: Dover Publications, 2016. ISBN: 978-0-486-80903-8.

[136] Mitchell Riley. *Categories of Optics*. Sept. 2018. arXiv: 1809.00738 [math.CT].

[137] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407. ISSN: 00034851. URL: http://www.jstor.org/stable/2236626.

[138] Mario Román. *Comb Diagrams for Discrete-Time Feedback*. 2020. arXiv: 2003.06214 [cs.LO].

[139] Robert Rosebrugh and RJ Wood. "Relational databases and indexed categories". In: *Proceedings of the International Category Theory Meeting 1991, CMS Conference Proceedings*. Vol. 13. 1992, pp. 391–407.

[140] H. H. Rosenbrock. *State-space and multivariable theory*. Studies in dynamical systems. Literaturverz. S. 252 - 254. New York, NY: Wiley, 1970. 257 pp. ISBN: 0471736457.

[141] John Rust. "Numerical dynamic programming in economics". In: *Handbook of Computational Economics*. Amsterdam: Elsevier, 1996, pp. 619–729. ISBN: 0-444-89857-3. DOI: 10.1016/s1574-0021(96)01016-7.

[142] Patrick Schultz, David I. Spivak, Christina Vasilakopoulou, and Ryan Wisnesky. *Algebraic Databases*. 2016. arXiv: 1602.03501 [math.CT].

[143] P. Selinger. "A Survey of Graphical Languages for Monoidal Categories". In: *New Structures for Physics*. Springer Berlin Heidelberg, 2010, pp. 289–355. ISBN: 9783642128219. DOI: 10.1007/978-3-642-12821-9_4.

[144] Dan Shiebler. "Categorical Stochastic Processes and Likelihood". In: *Compositionality* 3 (Apr. 2021), p. 1. ISSN: 2631-4444. DOI: 10.32408/compositionality-3-1.

[145] Michael A. Shulman. "Framed bicategories and monoidal fibrations". In: *Theory and Applications of Categories* 20.18 (2008), pp. 650–738. arXiv: 0706.1286 [math.CT].

[146] 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, and Demis Hassabis. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (Oct. 2017), pp. 354–359. ISSN: 1476-4687. DOI: 10.1038/nature24270.

[147] David Silver, Satinder Singh, Doina Precup, and Richard S. Sutton. "Reward is enough". In: *Artificial Intelligence* 299 (Oct. 2021), p. 103535. ISSN: 0004-3702. DOI: 10.1016/j.artint.2021.103535.

[148] Toby St Clere Smithe. "Mathematical foundations for a compositional account of the Bayesian brain". PhD thesis. University of Oxford, 2023.

[149] Sokolova, A (Ana). "Coalgebraic analysis of probabilistic systems". en. PhD thesis. Technische Universiteit Eindhoven, 2005. DOI: 10.6100/IR596314.

[150] Eduardo D. Sontag. *Mathematical Control Theory*. Springer New York, 1998. ISBN: 9781461205777. DOI: 10.1007/978-1-4612-0577-7.

[151] David I. Spivak. "Generalized Lens Categories via functors $C^{op} \to Cat$". In: (Feb. 2020). arXiv: 1908.02202 [math.CT].

[152] Mark W. Spong and Laurent Praly. "Control of underactuated mechanical systems using switching and saturation". In: *Control Using Logic-Based Switching*. Springer-Verlag, pp. 162–172. ISBN: 3540760970. DOI: 10.1007/bfb0036093.

[153] David Sprunger and Shin-ya Katsumata. "Differentiable Causal Computations via Delayed Trace". In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, June 2019, pp. 1–12. DOI: 10.1109/lics.2019.8785670.

[154] Robert F. Stengel. *Optimal control and estimation*. Dover Books on Mathematics. Description based upon print version of record. New York: Dover Publications, 1994. 11131 pp. ISBN: 9780486134819.

[155] Ross Street. *Monoidal categories in, and linking, geometry and algebra*. 2012. arXiv: 1201.2991 [math.CT].

[156] Kirk Sturtz. *Categorical probability theory*. 2015. arXiv: 1406.6030 [math.CT].

[157] Richard S. Sutton. "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, 1995, pp. 1038–1044.

[158] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. en. MIT Press, 2020, p. 352. ISBN: 9780262039246.

[159] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. "Policy gradient methods for reinforcement learning with function approximation". In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*. NIPS'99. Denver, CO: MIT Press, 1999, pp. 1057–1063.

[160] Dmitry Vagner, David I. Spivak, and Eugene Lerman. *Algebras of Open Dynamical Systems on the Operad of Wiring Diagrams*. 2014. arXiv: 1408.1598 [math.CT].

[161] Pietro Vertechi. "Dependent Optics". In: *Electronic Proceedings in Theoretical Computer Science* 380 (Aug. 2023), pp. 128–144. ISSN: 2075-2180. DOI: 10.4204/eptcs.380.8.

[162] Bret Victor. *Up and Down the Ladder of Abstraction*. Blog post. Oct. 2011. URL: https://worrydream.com/LadderOfAbstraction/.

[163] André Videla and Matteo Capucci. *Lenses for composable servers*. 2022. arXiv: 2203.15633 [cs.NI].

[164] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung 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, and David Silver. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (Oct. 2019), pp. 350–354. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1724-z.

[165] Robert F.C. Walters. "The free category with products on a multigraph". In: *Journal of Pure and Applied Algebra* 62.2 (Dec. 1989), pp. 205–210. ISSN: 0022-4049. DOI: 10.1016/0022-4049(89)90152-7.

[166] Christopher J. C. H. Watkins. "Learning from Delayed Rewards". PhD thesis. Cambridge, England: University of Cambridge, 1989.

[167] Jan C. Willems. "The Behavioral Approach to Open and Interconnected Systems". In: *IEEE Control Systems Magazine* 27.6 (2007), pp. 46–99. DOI: 10.1109/MCS.2007.906923.

[168] Donald Yau. *Operads of Wiring Diagrams*. Lecture Notes in Mathematics Ser. v.2192. Cham: Springer International Publishing AG, 2018. 1302 pp. ISBN: 9783319950013.

[169] Lotfi Asker Zadeh. *Linear system theory. the state space approach*. McGraw-Hill, 1963, p. 628. ISBN: 9780070727465.