On the Foundations of Universal Artificial Intelligence

Author:
Elliot Catt
Supervisors:
Prof. Marcus Hutter
Dr Joel Veness

A thesis submitted for the degree of Doctor of Philosophy

School of Computing

April 2022

Declaration

This dissertation is an account of research undertaken between July 2018 and April 2022 at the School of Computing, The Australian National University, Canberra, Australia.

The work presented in this thesis is that of the candidate alone, except where indicated by due literature reference and acknowledgements in the text. It has not been submitted in whole or in part for any other degree at this or any other university.

This thesis is based on three papers published in refereed scientific journals, two papers submitted for publication, one paper in preparation for submission, as well as one complete book which has been submitted for publication. The development of ideas and research was undertaken with guidance from my supervisors Marcus Hutter, Joel Veness and Laurent Orseau and published papers presented in the thesis were written collaboratively with them as well as other coauthors. Below I describe my contributions to the research in each chapter and associated book and papers.

- Chapter 2 is part of the forthcoming book Catt, Quarel et al. (2022) where my research forms a major contribution to all chapters of the book.
- Chapter 3 is the paper Catt and Norrish (2021) where my research forms a major contribution to all sections of the paper.
- Chapter 4 is the paper Catt, Hutter et al. (2022) where my research forms a major contribution to all sections of the paper.
- Chapter 5 is the paper Catt, Veness et al. (2022a) where my research forms a major contribution to all sections of the paper.
- Chapter 6 is the paper Catt, Veness et al. (2022b) where my research forms a major contribution to all sections of the paper.
- Chapter 7 is the paper Cohen, Catt et al. (2019). Section 7.5 is my major research contribution to the paper.
- Chapter 8 is the paper Cohen, Catt et al. (2021). Section 8.8 is my major research contribution to the paper.

Elliot Catt
1 April 2022

Acknowledgements

To Marcus, your constant aid and insightful advice has made me not just a better researcher and scientist, but a better human. You have supported me in so many ways, across so many endeavours. I cannot thank you enough.

To my wife Amy, I don't think anyone has read my work as much as you. Your constant feedback and support has made every aspect of my work easier. Thank you.

To Joel, I have so many things to thank you for. To start with thank you for your tireless supervision, which would often stretch out past work hours. Thank you for providing me with the environment which allowed me to work best. Lastly, thank you for being such a good friend.

To David, thank you for reading and helping to write the book, as well as providing clarity and comments which helped to enhance my own understanding.

To Michael N, thank you for spending the time you did with me; through our work together I was able to gain a much deeper understanding of many topics.

To Michael C, thank you for helping me, not just in Macao but in understanding AGI, safety and why Edric is too strong in Commander.

To Sultan, thank you for always being there for insightful discussions and thank you for all the feedback you have given me.

Matthew, Tom and everyone else that helped read and give various forms of feedback for my work, thank you.

To the Australian National University and all the employees that have helped make the processes of studying and completing my degree easier, thank you.

This work has been supported in parts by the Australian Research Council grant DP150104590.

Abstract

The goal of the field of artificial intelligence has always been the construction of a truly intelligent system. Understanding the overall behaviour and function of a potentially intelligent agent is key to its construction.

The leading theory in understanding the behaviour of artificial general intelligence is the theory of Universal Artificial Intelligence (UAI). According to this theory, intelligent agents will act optimally with respect to a universal Bayesian perspective. As part of this work, we have produced a textbook detailing the study and development of the field of UAI since its inception 22 years ago.

One key component of the universal Bayesian setup is the universal prior which is a belief that simple explanations are more likely correct than complex explanations. Mathematically this prior is characterised by the Kolmogorov complexity, which is a well-studied measure of simplicity that has many ideal properties. In this thesis, we formally verify many of its properties through the use of mechanisation in the HOL4 theorem prover. With this formal verification, we are able to attain a greater sense of certainty about the validity of Kolmogorov complexity as a measure of Complexity and deepen our understanding of it through new proof techniques.

When acting in an environment, agents use many distinct action spaces. A chess-playing agent has the set of legal moves, a car-driving agent uses steering, acceleration and braking, and a programming agent considers the set of possible programs it can write. In this thesis, we present an information-theoretic method in which all action spaces can be unified into a single succinct action space, with the aid of recent advances in large language models. We show that under this method, the traditional setup of an agent interacting with an environment is preserved.

We go on to apply this action unification technique to a policy evaluation approach, called compress and control, which is able to take advantage of the unified structure of the actions. We show that under the action unification setup, compress and control leads to a policy evaluation that is a consistent estimator of the true value of the policy.

In the UAI framework, one of the "free variables" is the choice of reward space. What we mean by this is that the choice of reward space is left as an unspecified implementation detail. In this thesis, we show that the learning ability of a Bayesian agent is independent of the choice of reward space. Specifically, we demonstrate that an agent with a binary reward space can learn just as well as an arbitrary reward space. This result shows that while the choice of reward space is free, it will not have an effect on the limiting performance of the agent.

While it can be shown that the Bayesian agent AIXI satisfies the conditions of Bayesian optimality, there are many distinct possible definitions of optimality in the UAI framework. These include (but are not limited to) regret minimisation, (weak/mean/strong) asymptotic optimality, Pareto optimality, and ν-optimality. Some of these choices for optimality are too strong to achieve, such as ν-optimality, and some are too weak to be useful, such as Pareto optimality. In the case of strong asymptotic optimality, it has been previously shown that no deterministic agent can achieve this. In this work we describe a stochastic agent, called Inq, which we prove is strongly asymptotically optimal. Additionally we demonstrate the performance of Inq experimentally, comparing it to BayesExp, a weak asymptotically optimal agent, and Thompson sampling, an agent which is asymptotically optimal in mean.

In this thesis we show that if an agent is asymptotically optimal, then it will eventually be incapacitated, essentially it will die. We go on to define an agent Mentee which, while not asymptotically optimal, explores according to a safe (human) Mentor policy and learns to eventually outperform the Mentor while asking for advice less as time goes on. We also show that experimentally, with the help of the Mentor's safe exploration, the Mentee agent can outperform asymptotically optimal agents as they will often fall into traps.

Keywords: artificial general intelligence; algorithmic information theory; rational agents; sequential decision theory; universal intelligent agents; reinforcement learning; AGI safety.

Contents

Figures

Tables

Abbreviations

RL	**R**einforcement **L**earning
AI	**A**rtificial **I**ntelligence
UAI	**U**niversal **A**rtificial **I**ntelligence
AGI	**A**rtificial **G**eneral **I**ntelligence
MDP	**M**arkov **D**ecision **P**rocess
POMDP	**P**artially **O**bservable **M**arkov **D**ecision **P**rocess
LLM	**L**arge **L**anguage **M**odel
MCTS	**M**onte **C**arlo **T**ree **S**earch
UCT	**U**pper **C**onfidence **T**ree
IG	**I**nformation **G**ain
KL	**K**ullback-**L**eibler **D**ivergence
KC	**K**olmogorov **C**omplexity
SI	**S**olomonoff **I**nduction
AIT	**A**lgorithmic **I**nformation **T**heory
SDT	**S**equential **D**ecision **T**heory

To my wife.

Chapter 1

Introduction

The long-standing goal of the field of Artificial Intelligence (AI) is the creation of a general problem solver, often called Artificial General Intelligence (AGI). To this end, the theory of Universal Artificial Intelligence (UAI) was formally developed. One of the most important components of the theory of UAI is the agent AIXI. AIXI is the most intelligent (in a sense) agent possible, however it is incomputable. AIXI, and UAI as a whole, builds on Sequence Prediction (SP), Algorithmic Information Theory (AIT) and Sequential Decision Theory (SDT), and Reinforcement Learning (RL).

In this chapter I will outline my thesis statement, go over the progress and contributions that have been made for the field of UAI within this thesis, and provide an outline to the chapters of this thesis.

1.1 Thesis Statement

We can summarise this thesis in the following thesis statement:

> *Universal Artificial Intelligence is a viable and realistic approach to the problem of artificial general intelligence.*

That is to say that, even though the most intelligent agent, AIXI, is incomputable, if we continue to improve various approximations of AIXI, we will eventually reach practical and implementable artificial general intelligence. Additionally, any theoretical results about AIXI, its components and other universal agents will prove instrumental in this development as they can help to guide us on the path towards artificial general intelligence.

1.2 Contributions/Outline

We provide several contributions to the field of UAI. These include: A forthcoming textbook on the topic of UAI; formal verification of Kolmogorov complexity, a key component of the UAI framework; determining the existence of a universal action space and showing how this universal action space can be used in practice; determining the existence of a universal reward set; experimentally demonstrating the viability of a strongly asymptotically optimal agent; and experimentally demonstrating the viability of an agent which safely explores.

Apart from Chapter 1 and Chapter 9, each chapter is based on a piece of work that is either published or currently in the publication process. Specifically,

- In Chapter 2 we present an adapted introduction of the forthcoming book Catt, Quarel et al. (2022), which describes the complete works of the theory of UAI.
- In Chapter 3 we present the formalisation of Kolmogorov complexity and its related theorems. Based on Catt and Norrish (2021).
- In Chapter 4 we go over an information-theoretic approach to the action space problem that leads to a universal action space in reinforcement learning. Based on Catt, Hutter et al. (2022).
- In Chapter 5 we expand on the results from Chapter 4 and apply them to the Compress and Control policy evaluation algorithm. Based on Catt, Veness et al. (2022a).
- In Chapter 6 we show that using only binary rewards can be sufficient for representation in reinforcement learning in many situations, however, there exist situations wherein binary rewards cannot capture the required information. Based on Catt, Veness et al. (2022b).
- In Chapter 7 we develop the first universal agent which is shown to be strongly asymptotically optimal. An implementation of this agent is able to outperform other known asymptotically optimal agents. Based on Cohen, Catt et al. (2019).
- In Chapter 8 we show that any agent which is asymptotically optimal will eventually become incapacitated. We define an agent which safely explores, called Mentee, and show that with this safe exploration Mentee can experimentally outperform other universal agents. Based on Cohen, Catt et al. (2021).
- In Chapter 9 we conclude the thesis and provide some potential future areas of work.

Chapter 2

An Introduction to Universal Artificial Intelligence

This chapter contains the first chapter of the forthcoming book

E. Catt, D. Quarel et al. (2022). *An Introduction to Universal Artificial Intelligence*. The draft version can be found at http://www.hutter1.net/ai/uaibook2.htm.

Comments

One of the major projects taken part in this thesis is the completion of a textbook on the topic of Universal Artificial Intelligence. Since the inclusion of a complete book would be too much for this thesis, we have instead just included an adapted first chapter; the full book can be found with the above link.

2.1 Introduction

Humans have been learning and copying from nature since time eternal. The airplane would hardly have been conceived as a possibility had we not first seen animals capable of flight. The components for a camera mimic that of the eye: The aperture of the camera acts as the pupil does to moderate the amount of light let in, and the photosensitive sensor at the back of the camera acts as the retina does. Even the humble hook-and-loop fastener was inspired[1] by the hooked barbs of a thistle bush. We don't just copy what nature has directly provided, but we learn the rules by which she works, so that we may improve her for our own needs: We domesticate wolves into dogs and selectively breed fruit and vegetables for a higher yield. As our understanding of nature advances, we have learned how to artificially fertilise soil, and how to directly edit the genetic code of organisms to take on properties that we desire. Artificial muscles built from steel never tire, and have freed countless people from the drudgery of repetitive hard labour.

How is it that humanity can perform such miracles? At first it's merely recognition of patterns: Humans discovered that crop rotation improved yields at harvest, or that selectively breeding animals with desired traits improved upon those traits, well before it was understood why these approaches worked. Later on, we constructed theories and laws to help explain the observations of the world around us, and make predictions based on those laws. We study natural phenomena, and then harness them for our own ends. This acts as a positive feedback loop: The more our knowledge is extended, the better equipped we are to advance it further. Innovations with agriculture allowed people to specialise in other skills, or devote a lifetime to furthering knowledge through literature, philosophy, science and mathematics. The written word allows transmission of ideas and concepts at an unprecedented bandwidth, allowing a conversation with some of the greatest minds in history beyond the grave. Calculus, once a branch of mathematics that only a few dozen people were skilled practitioners in, is now taught to most teenagers as part of their standard curriculum in school.

All of these innovations share a common source, and that source is what makes humanity unique among other animals: our high level of intelligence, the ability for us to reason, strategise, and plan ahead, taking actions to bring the state of the world into one more desirable to us. We would like to do to the brain what the combine harvester did to the oxen: To build an artificial version that can think and plan for us, and further satisfy our values. Intelligence is one of the remaining unsolved phenomena that we still know very little about. How does the brain, a kilogram or so of meat through which electrical impulses fire, give us our intelligence, our consciousness, and a personality? The field of neuroscience is still very new, and has categorised portions of the brain based on the parts of the body they interface with (vision, speech, hearing) or the tasks by which the brain can function to control the body (memories, reflexes, emotions) but provides no answer to the questions presented.

While there have been many (successful) attempts at the construction of *narrow AI* (an AI that performs well in a single or narrow class of domains, such as Chess), no-one has yet

[1] or plagiarised, from the point of view of the thistle bush

succeeded in creating an AI which is able to match or exceed human intelligence *in a wide class of environments*, often called Artificial General Intelligence (AGI). Before we can build an AGI, arguably we must first develop understanding of what AGI is. How could we possibly hope to understand something as daunting and complex as AGI? The same way we are able to understand other complicated phenomena such as the transmission of genes, or the processes that fuel the stars: through the construction of a formal mathematical theory through which properties about the phenomena can be derived. At its core, that is the purpose of the book Catt, Quarel et al. (2022): to construct, explain, and demonstrate the formal theory of *Universal Artificial Intelligence* (UAI).

While we may not know the internal structure that a hypothetical future AGI would possess, we are primarily concerned with theory regarding the *behaviour* of such intelligent agents[2] in the hope that practical realisations of intelligence can be constructed upon this theory. We didn't get to the moon by building lots of different rockets in the hope that one might work, but by first creating idealised models to see if such a task is even feasible. All the theory should be well developed before the first rocket is even constructed. At the time of writing, AGI is still very much in the theory stage, which is where we will focus on in the book Catt, Quarel et al. (2022).

It may be difficult to believe that there exists a theory which is expansive enough to capture all the intricacies that intelligent behaviour includes. However, we will show that UAI encapsulates any reasonable definition of intelligence and intelligent behaviour. In addition, we will show that within this theory there exists an agent that can be shown to be in a sense "maximally intelligent", known as *AIXI*.

The book Catt, Quarel et al. (2022) is meant to serve as an introductory text to the formal treatment of the topic of artificial general intelligence, as well as a summary of the work done so far.

2.2 Overview

Much of UAI builds upon and uses results from many already established theories. These include (but are not limited to) Bayesian probability theory, decision theory, computability theory, complexity theory, information theory, artificial intelligence, algorithmic information theory, sequence prediction, sequential decision theory, reinforcement learning and game theory. We have included a background chapter (Catt, Quarel et al., 2022, Chapter 2) (together with suitable auxiliary sources) to help the reader with the prerequisite material that this book builds upon.

[2]Our lack of understanding precisely how the human brain works hasn't stopped us from acting intelligently, or the field of psychology from trying to model human behaviour.

2.3 Algorithmic Prediction

Prediction is the ability to accurately estimate the outcome of future events based on the results from past events. A good predictor should be able to work out which of many possible futures is the most likely. When we formalise this in mathematics, we will often describe it as predicting the next element (or elements) in a given sequence. For example, given the sequence

$$3,\ 1,\ 4,\ 1,\ 5,\ 9$$

a predictor would be tasked with determining the next element.

Prediction is at the heart of any intelligent behaviour. Whether this prediction is explicit or implicit, prediction is required for intelligence. It is important to note that (good) prediction is necessary but not sufficient for intelligence. This is because (under how we define intelligence) intelligence is measured as a function of the actions the agent takes, whereas prediction assumes that the predictions made do not affect the environment.

Returning to the above sequence, how do we determine the next element? The observant reader may have noticed that this sequence matches the first six digits of π in base 10. With this information we could predict the next element to be the seventh digit of π in base 10, the number 2. This is likely to be a good prediction, however, digits of π in base 10 is not the only possible continuation. There are infinitely many other sequences that start with $3, 1, 4, 1, 5, 9$; one such example would be the decimal expansion of $355/113$, which is arguably as simple as the circle constant π. Why should we prefer the sequence continuation to be the digits of π over that of $355/113$, or any other valid continuation? According to *Epicurus' principle of multiple explanations*,

If more than one theory is consistent with the observations, keep all theories.

So, we cannot discount the possibility that the decimal expansion of $355/113$ (or something else) is the correct continuation. But this does not mean we should consider all possible continuations as being equally likely to be the true answer. Here we can employ the most important philosophical tool for science, *Occam's Razor*, which states

Entities should not be multiplied beyond necessity.

which can be interpreted as

Keep the simplest theory consistent with observations.

So in our above example we consider the possibility that any number could be next in the sequence, but we bias ourselves towards more simple explanations[3], in this case biasing

[3] Is π a simple explanation? What about $355/113$? Which is "simpler"? How can we formally define "simplicity"? We investigate this later.

ourselves towards the number 2. More data will soon reveal the distinction between π and 355/113, and the hypothesis inconsistent with the data can be ruled out.

Following the above principles, we do consider all possible sequences in some reference class[4], discard those inconsistent with the data so far, and bias our prediction toward the simplest[5] sequences. Given our current belief of which continuation of the sequence is the most likely, how should we refine this belief when more evidence is observed? For instance, in the above example if we found out the next element was not 2 as expected, but 6, then how should this new information be incorporated to our prediction? We let O denote the set of all possible outcomes (sequences) and E denote the set of all pieces of evidence (finite prefixes of sequences) that can be observed that can be observed. Given some *prior* $P(O = o)$ (the confidence we have that o is the true outcome before any evidence was observed) and a piece of evidence $e \in E$, we can mathematically express what our new confidence in o should be when given evidence e (the *posterior* $P(O = o|E = e)$) in terms of the *prior*, and how likely it would be that evidence e would be observed given that o is the true outcome (the *likelihood* $P(E = e|O = o)$) as follows:

$$\begin{aligned} P(O = o|E = e) &= \frac{P(E = e|O = o)P(O = o)}{P(E = e)} \\ &= \frac{P(E = e|O = o)P(O = o)}{\sum_{o' \in O} P(E = e|O = o')P(O = o')}. \end{aligned}$$

This relationship is called *Bayes' Law*. The only thing left to specify is the choice of *prior* $P(O = o)$, which is itself an entire topic of debate. Using *Occam's razor*, we can bias the prior towards more "simpler" outcomes. This can be done formally by building upon material from computability theory, and was neatly combined into a single formal theory of inductive inference known as *Solomonoff Induction*, by Ray Solomonoff (Solomonoff, 1964a).

Solomonoff Induction

The sequence prediction problem has been solved (theoretically) by Solomonoff Induction (SI) (Solomonoff, 1964a,b), which is based on the Solomonoff Prior:

$$M(x) := \sum_{p:U(p)=x*} 2^{-\ell(p)}.$$

Here $U : \{0,1\}^* \to \{0,1\}^*$ is a (monotone) Universal Turing Machine (UTM), $x*$ is any binary string starting with x, and $\ell(p)$ is the length of a binary string p. The Solomonoff Prior is the probability of outputting a string x if a uniformly random input is given to the UTM U.

[4] We cannot possible consider all possible sequences that have $3, 1, 4, 1, 5, 9$ as a prefix, as there would be unaccountably many sequences to consider. We also wouldn't want to consider a class too restrictive, as if it were the case that π were not in the reference class, then we would never learn the true sequence. A solution to this will be discussed later.

[5] *Simplicity* will be defined formally later on.

Before we get to how Solomonoff's a-priori distribution can be used to solve the sequence prediction task, we need to define Kolmogorov Complexity (KC). The KC of a binary string is the length of the shortest program which can compute it on universal Turing machine U, and infinity if no such program exists. Formally,

$$K(x) = \min_{p \in \{0,1\}^*} \{\ell(p) | U(p) = x\}.$$

At first glance KC seems similar to the Solomonoff Prior, and in fact they are similar. We have the relation $M(x) \approx 2^{-K(x)}$; since the program in KC is the smallest program which computes x, it will dominate the sum in the Solomonoff Prior. It is important to note that both the Solomonoff Prior and KC are incomputable.

How does Solomonoff Induction solve the sequence prediction task? We need to use the conditional Solomonoff distribution $M(x|y) = \frac{M(yx)}{M(y)}$. Using this we have the following theorem Solomonoff (1964a,b):

Theorem 2.1 (Solomonoff (1985))**.** *If μ is a computable measure and $x = x_1 x_2 \ldots$ is distributed according to μ, then*

$$\sum_{n=1}^{\infty} \sum_{x:\ell(x)=n-1} \mu(x) \left(M(0|x) - \mu(0|x)\right)^2 \leq K(\mu) \frac{\ln(2)}{2} < \infty$$

where $K(\mu)$ is the KC of μ.

This theorem implies $M(x_{t+1}|x_1 x_2 \ldots x_t) \to \mu(x_{t+1}|x_1 x_2 \ldots x_t)$ with μ-probability 1. This means that for any sequence which has a computable source, SI will converge to the correct predictions, and converge quickly.

The theory of SI can then be used for prediction, but it comes with some drawbacks. Most notably, the method by which the predictions are evaluated under this theory under the choice of Solomonoff's *universal prior* are incomputable, meaning that there exists no algorithm by which the predictions can be computed. With such a drawback, we might question what possible use algorithmic probability has. Algorithmic probability is still useful as measure of what an (impossibly) good predictor looks like, and variants can be constructed that weaken the strength of the predictions (by choosing an easier space of outcomes, or a sub-optimal prior) but also bring them back to something for which an algorithm exists. One such method is the *Context Tree Weighting* (CTW) predictor, which uses as its reference class the set of all *k-Markovian*[6] models, and uses a tree structure to efficiently update the belief distribution based on symbols observed. (Catt, Quarel et al., 2022, Chapter 4). Like Solomonoff Induction, the Bayesian mixture provided by CTW uses a prior based on simplicity[7], which was shown to be a viable algorithm for prediction, with both good theoretical bounds on performance,

[6]The probability distribution over the next symbol is a function solely of the last k symbols.

[7]each hypothesis in the model class can be represented by a binary tree (Catt, Quarel et al., 2022, Section 4.4) and the prior is related to the length of an encoding of the tree as a binary string (Catt, Quarel et al., 2022, Definition 4.28)

(Catt, Quarel et al., 2022, Section 4.6.1) as well as demonstrating strong powers of prediction in practical experiments. (Catt, Quarel et al., 2022, Section 4.4.3).

There are several extensions of Context Tree Weighting which are able to efficiently predict related (or in some cases larger) model classes with comparable efficiency to traditional context tree weighting. These include methods such as *Context Tree Switching* (Catt, Quarel et al., 2022, Chapter 5), which includes models that switch between different distributions depending on the context seen[8], instead of a single model which depends on the context.

Additional Literature. Since its inception, SI has been applied to different problems including AI (Solomonoff, 1985), Machine Learning (ML) (Solomonoff, 2003) and many more. For a thorough explanation of KC and SI, Li and Vitányi (2008) is recommended. For many applications of Algorithmic Information Theory, see Li and Vitányi (2007). For the philosophical arguments that SI is the solution to the induction problem, the paper Rathmanner and Hutter (2011) is recommended; it presents and discusses SI and its many aspects in a very readable manner, without dense mathematical notation.

Solomonoff Induction and AIT are key components of UAI, and there have been several developments on these topics and the topic of sequence prediction. One drawback (or benefit) of the Solomonoff Prior is that it is a semimeasure, not a complete measure on binary strings. However it can be normalised to become a measure on binary strings. It has been shown that this normalised version of the Solomonoff Prior, denoted by M_{norm}, is stronger than M (Lattimore, Hutter and Gavane, 2011) in the sense that there are sequences which M_{norm} can predict that M cannot.

A universal theory for prediction and confirmation which may be used on any inductive inference problem was developed in Hutter (2007). In doing so, the authors showed the benefits and pitfalls of using the universal theory for prediction, specifically how it solves many of the difficult problems in the Bayesian framework and SI. Martin-Löf random sequences are an important part of Algorithmic Information Theory; a sequence is said be be incompressible if it is Martin-Löf random. It has been shown that there exist Martin-Löf random sequences such that there is a universal semimeasure which does not converge to the sequence. However, on a more positive side it has been shown that there exist (potentially non-universal) semimeasures which converge to μ on all Martin-Löf random sequences (Hutter and Muchnik, 2007). Continuing to consider random sequences, it has been shown that there exists a Lebesgue random string such that the limit of the conditional distribution of any universal mixture does not predict the next element with probability a half.

An important distinction in prediction problems is that between offline and online prediction. In offline prediction, the goal is to learn a probability distribution from a batch of data. In online prediction the goal is, as we have described in this section, predicting the next element in a sequence given the previous elements. Converting an offline predictor to an

[8] Useful for predicting sequences where a long run of contiguous terms are generated from a hidden distribution, and the distribution then changes abruptly at some point to a different distribution (e.g. when compressing a file, the border between text data and image data).

online predictor efficiently is non-trival, however Hutter (2014) proposed an efficiently-timed converter and provided some results on the minimal regret achievable.

A type problem that occurs often in prediction is that of non-stationary sources. There are many approaches to deal with this type of problem, the Context Tree Switching (Veness, Ng, Hutter and Bowling, 2012) mentioned earlier is one method. Another method is to use discounting. It has been shown that for sequences of important classes that do not drift too much, discounting works well (Sunehag, Shao et al., 2012). The idea of discounting has been applied to the CTW algorithm (Hutter, Shao et al., 2012), allowing CTW to work for non-stationary sources. The discounting applied to CTW slightly outperformed the original CTW on test sets.

2.4 A Family of Universal Agents

Intelligence is more than being able to predict well; intelligence requires being able to act well. Acting well could mean many different things:

- Achieving some predetermined goal.
- Maintaining some objective.
- Minimising a loss function.
- Respecting a set of constraints.
- Acting *rationally*[9].
- Making correct decisions based on information present.

We can capture all of these ideas within a single framework, known as the *cybernetic model*.[10] In this framework an agent takes actions and receives both observations and rewards from an environment, and the goal of the agent is to maximise the expected reward from the environment. The intelligence of humans is merely a byproduct of an evolutionary arms-race optimising for reproductive fitness (the smarter you are, the better you can seek food, avoid predators, and survive to bear children who inherit your genes). Evolution has baked into humans the "reward signals" of pleasure and pain which are activated when taking actions that are correlated/uncorrelated with reproductive fitness (eating, mating, feeling safe, making friends is pleasurable; physical damage, social ostracism and hunger is painful). It stands to reason that this might also be possible with virtual agents seeking to maximise a reward signal of our design to incentivise the behaviours we desire.

The repeated interaction is performed by the agent taking an action from a finite action set $\mathcal{A}$, then the environment providing an observation o and reward r from countable sets $\mathcal{O}$

[9] . . . whatever that means.

[10] Also called *general reinforcement learning* or *history-based reinforcement learning*. Many formal presentations of reinforcement learning assume that the agent taking actions, or the environment with which they interact (or both) are *Markovian*, in that their behaviour is dictated only by the current state, and not from observations in the past. Obviously, a predictor can only hope to predict Markovian sequences if it can only remember the previous symbol.

and $\mathcal{R} \subset [0,1]$ respectively. The set of all possible observations is $\mathcal{H} = (\mathcal{A} \times \mathcal{O} \times \mathcal{R})^* = \cup_{t=0}^{\infty} (\mathcal{A} \times \mathcal{O} \times \mathcal{R})^t$. The agent $\pi : \mathcal{H} \to \Delta\mathcal{A}$ is a stochastic function[11] from histories $h_{<t} = a_1 o_1 r_1 \dots a_{t-1} o_{t-1} r_{t-1} \in \mathcal{H}$ to actions. An environment $\nu : \mathcal{H} \times \mathcal{A} \to \Delta(\mathcal{O} \times \mathcal{R})$ is a stochastic function from histories and an action to an observation and reward. The goal of the agent is to choose actions which maximise reward. Instead of trying to maximise the immediate reward, in RL the agent also has a discount function γ_t and the goal is to maximise a value function which includes all future reward. This value function at time t is defined by

$$V_\nu^\pi(h_{<t}) := \frac{1}{\Gamma_t} \mathbb{E}_\nu^\pi \left[\sum_{k=t}^{\infty} \gamma_k r_k \middle| h_{<t} \right]$$

where $\Gamma_t := \sum_{k=t}^{\infty} \gamma_k$ is a normalisation constant based on the discounting function, and r_i is the reward at the ith timestep.

The actual goal that we want the agent to achieve can be encoded in the rewards issued. If we want the agent to win games of chess, we can issue positive rewards for when it wins a game, and negative rewards when it loses a game. An agent trying to maximise the expected sum of rewards will have to learn to play well at chess.

As we will see (Catt, Quarel et al., 2022, Chapter 6), with this setup we can describe all of the above ideas in one unifying fashion, through judicious choice of the environment with which the agent interacts.

If the agent is aware of which environment it interacts with, then it can (in principle) deduce the best course of action (given no constraints on computational resources). For example, the game of chess can be solved at least in theory, as one can use the *minimax* algorithm from the initial board state to find out who wins given perfect play on both sides. While this would be intractable in practice, we can at least say that there exists an algorithm for determining the optimal action in Chess.

However, it turns out that in most cases the agent does not know which environment it is in and must learn from experience. We wish to describe a method by which an agent can interact with an unknown environment, learn what environment it is in via experience, and learn to act optimally.

Consider the case of someone playing a video game they have never played before. Initially the player will likely not play the game well, as they don't understand the mechanics of the game, and may make either sub-optimal or illegal[12] moves as they do not fully understand the rules of the game (they do not know which environment they are in). Over time they will learn the rules of the game and narrow down the possible environments they are in until they fully understand the rules and dynamics of the game (environment), and can then try to act optimally.

[11] A stochastic function, or conditional distribution f from A to B, denoted $f : A \to \Delta B$, is a function from A to the set of all probability distributions on B, denoted ΔB. If $f(a) = p_a$ for some probability distribution $p_a \in \Delta B$, then we define $f(b|a) := p_a(b)$, the probability that the stochastic function f returns b given a as input.

[12] Illegal moves may be possible in board games, but may be impossible in a video game, as the game will enforce the rules for the player and prevent them from taking actions that break the rules of the game.

Initially, the agent knows nothing of the true environment. We make the (weak) assumption that the environment with which the agent interacts is computable, so by taking the set of all computable environments as the reference class of environments, we have that the true environment will be contained in the agent's reference class.[13]

Solomonoff induction then allows us to determine the probability we are in a given environment, as predicted by an optimal predictor. Putting all these ideas together, we are able to arrive at the most intelligent environment-independent agent, also known as the Bayes optimal agent, AIXI[14]:

$$\text{AIXI}(h) := \arg\max_{\pi} V_{\xi}^{\pi}(h)$$

where we define the mixture of environments ξ as $\xi(x) := \sum_{\nu \in \mathcal{M}} \nu(x) w_\nu$, where $w_\nu = 2^{-K(\nu)}$ and $\mathcal{M}$ is the space of (semi)computable (semi)measures. From a philosophical perspective, like SI, AIXI takes from Epicrus' principle of multiple explanations, through the mixture ξ considering all environments, and Occam's Razor, through the weighting of each environment in ξ by its complexity $w_\nu = 2^{-K(\nu)}$. It has been shown that for every environment ρ in AIXI's model class, if there exists an optimal agent π^ρ for ρ, AIXI will eventually act as well as π^ρ. Essentially, whatever optimal behaviour looks like in an environment, then AIXI will (eventually) act optimally. This, coupled with the fact that AIXI is the optimal Bayesian agent with the universal prior, demonstrates that AIXI is the theoretically most intelligent agent. Of course, due to the components of AIXI being incomputable, AIXI is also incomputable, but it can serve as a gold standard in the construction of more practical paths to AGI, much like the intractable algorithm of exhaustive minimax for Chess paved the way for practical realisations of Chess bots.

While Bayes optimality is arguably a sensible choice of optimality criterion for agents in the UAI framework, there are several alternative optimality criteria for intelligent agents, each with their upsides and downsides. The agent AIXI has been extended in several directions to both deal with these alternative criteria, and answer some of the remaining open questions in the construction of AIXI. One example of these is the Knowledge Seeking Agent (KSA) (Catt, Quarel et al., 2022, Section 9.4). The knowledge seeking agent behaves like AIXI, however instead of maximising the rewards issued from the environment, it replaces this with expected future information gain, that is, it takes actions that result in the most "surprising" observations from the environment.[15] In the book Catt, Quarel et al. (2022) we will go into full detail of all the AIXI variations and extensions, as well as their potential satisfaction of the various optimality criteria (Catt, Quarel et al., 2022, Chapter 9).

Additional Literature. There has been much work in the literature on extending the understanding of AIXI, overcoming the shortcomings, and modifying AIXI to new agents. Often

[13] For more about this assumption, see Catt, Quarel et al. (2022, Section 17.2)

[14] See Catt, Quarel et al. (2022, Chapter 7) for more details.

[15] While encouraging exploration, KSAs suffer from what is known as the *noisy TV problem*: A KSA stumbling across a source of random noise in the environment (like an old style analog television set tuned to an empty channel) would be absolutely delighted by what it sees and would remain transfixed, addicted to the constant supply of surprising observations.

the goal of this extension is to give it more exploration. Here we will describe some of that work.

To start with, we will consider the case that the chosen prior is static (history-independent). It was shown that if this is the case, then AIXI cannot converge to the optimal AIμ for all computable environments (Orseau, 2010). This was done by construction by defining a pair of environments in which AIXI will take the optimal action in one, and AIμ will take the optimal action in the other, and these are different actions. The author argues that this is a problem with any universally "greedy" agent like AIXI and some exploration is required. One method to achieve this exploration is with a knowledge seeking agent mentioned earlier. Another negative result in the UAI framework is that under certain conditions on the environment and the policy, asymptotically optimal policies, both strong and weak, are not possible (Lattimore and Hutter, 2011). The original knowledge seeking agent (Orseau, 2014) was extended to the stochastic setting and found to be robust to noise (Orseau, Lattimore et al., 2013). The KSA agent was then combined with the "greedy" AIXI to lead to an agent, known as BayesExp, which is weakly asymptotically optimal, avoiding the problems mentioned earlier (Lattimore and Hutter, 2014a). A compilation of the above results, and others, can be found in the thesis Lattimore (2014).

An alternative solution to the problem of asymptotic optimality was proposed was an optimistic agent which used the the rationality axioms and was shown to be asymptotically optimal (Sunehag and Hutter, 2015). In the deterministic case, the environments were decomposed in sub-environments called laws.

It has been shown that some of the desirable properties of AIXI (optimality, Pareto optimality, etc) are either trivial or highly dependent on the prior, and hence dependent on the the UTM used for the Kolmogorov complexity (Leike and Hutter, 2015). Again, this is a result of the lack of exploration which AIXI undertakes. There are some ways around this, such as exploring like the previously mentioned BayesExp, or having some "good" initial history.

Although it is known that Solomonoff's prior and AIXI are incomputable, the exact (in)-computability level for Solomonoff's prior (normalised and unnormalised), AIXI, AIν, and AIμ were shown in Leike and Hutter (2018), determining where they are on the arithmetic hierarchy. It was also shown that the recursive definition of the value function is objectively superior to the iterative definition. The authors also show that an epsilon-approximation of a policy is one level lower in the arithmetic hierarchy.

An excellent summary of the more theoretical work of UAI is the thesis Leike (2016). The author discusses notions of the optimality of agents in RL, showing that many optimality conditions do not make sense for various reasons. The author then presents an AIXI variation, Thompson sampling, and shows that it is asymptotically optimal in mean. Additionally the a solution to the grain of truth problem is presented; the grain of truth problem is the problem of finding a large class of policies which contains the Bayes-optimal policy with respect to the class.

2.5 Approximating Universal Agents

We cannot compute the optimal AIXI agent directly, but we can still try to implement weaker forms that approximate AIXI. There exist several such approximations of AIXI that we discuss here. Unsurprisingly, the "better" the approximation is, the harder it is to compute.

AIXI-MDP considers the class of possible environments to be the set of Markovian environments.[16] While AIXI-MDP is computable and can perform well in Markov environments, obviously it should and does struggle on more complex environments, as it can only at best learn the closest Markov approximation to the environment.

MC-AIXI-CTW (Catt, Quarel et al., 2022, Chapter 12) uses the context tree weighting mentioned earlier to efficiently update its belief in each possible environment, using the set of all k-Markov environments as its reference class. This, combined with Monte-Carlo Tree Search for planning (Catt, Quarel et al., 2022, Section 12.3), leads to an agent which is able to learn and perform well on complex environments.

The AIXI agent uses a *model-based* approach to the general reinforcement learning problem, in that it learns a potential model of the environment, and then uses that model as a substitute for the true environment when planning forward in the future.[17] We are not, however, interested in the agent which is the best at modelling the environment; we require an agent which is able to perform the best.

From our perspective, it doesn't matter if the agent attempts to construct a model of the environment, and in fact it should only spend resources doing in so far that it results in better performance overall, and the resources would not be better spent doing something else that results in better performance. One interesting *model-free*[18] approach is *compress and control* (CnC) (Catt, Quarel et al., 2022, Section 13.2). In CnC, the agent learns to model not the environment, but the expected reward sum (value) (Catt, Quarel et al., 2022, Chapter 6.4) directly. This approach relies upon a method to decompose the value in such a way that allows for efficient representation using information-theoretic techniques.

The final approximation that we will consider is a resource-constrained version of AIXI, known as AIXItl. This approximation of AIXI runs in time $O(t)$, uses space $O(l)$, and converges to AIXI as t and l go to infinity.

Additional Literature. One key component of building (direct) AIXI approximations is the model class. We cannot use the full model class that AIXI considers, however we would like to use a model class that is sufficiently large/general, while still being tractable. Various techniques for approximating large model classes - including weighting/averaging,

[16] That is, the set of environments where the probability of the next observation and reward depends only on the previous observation and action.

[17] Humans often do the same thing, in adversarial games like chess, a player might try to estimate the responses an opponent would make, and how they would respond in turn. When driving to work in the morning, we have a reasonably good estimate as to how other drivers on the road will act, and they have a reasonably good estimate of us.

[18] A method that does not explicitly construct a model of the true unknown environment

switching/tracking, convex mixing, and second order methods - have been discussed in Veness, Sunehag et al. (2012). All the proposed methods ran in polynomial time (several were linear) in the size of the model class.

Not an approximation of AIXI, but an approximation of the Intelligence Measure Y was defined in Legg and Veness (2011). The authors use an interesting sampling approach to approximate the universal intelligence measure Y of different agents. Using the BF language as the reference machine, which only has 8 symbols, the method was semi-practical and demonstrated the intelligence of certain agents such as MC-AIXI-CTW and Q-learning.

We have been describing various methods to build approximations for the agent AIXI, however, we have also described a family of (theoretical) universal agents including KSA and BayesExp. Approximations to the family of universal agents have been constructed and used to compare between them experimentally (Aslanides et al., 2017). The approximations relied on the MCTS used in Veness, Ng, Hutter, Uther et al. (2011), and the tests comparing them were done in small grid-world environments.

2.6 Alternative Approaches

The Bayesian approach of Universal Artificial Intelligence is not the only attempt to solve the general reinforcement problem. Of the other approaches, one of the more studied is *feature reinforcement learning* (Catt, Quarel et al., 2022, Chapter 15).

At an abstract level, feature reinforcement learning is similar to universal artificial intelligence in that it consists of a learning component and a planning component. In feature reinforcement learning, the agent learns to map the general (history-based) reinforcement learning problem to a more tractable problem (e.g. Markov Decision Process), and then plans on that more tractable problem. Feature reinforcement learning has yielded many promising methods for translating the hard problem of general reinforcement learning to more easily solvable problems. These include both theoretical results demonstrating the viability of FRL, and practical algorithms which can perform comparably to AIXI approximations like MC-AIXI-CTW. The practical algorithm was a combination of context tree maximizing (CTM) and RL called CTMRL (Nguyen et al., 2012). It was shown that compared to MC-AIXI-CTW the time required for CTMRL was drastically less.

2.7 Safety and Discussion

Is it safe to build an AGI? Safety for AGI is entirely unlike managing the safety of other dangerous things, like nuclear weapons or highly infectious diseases. Nuclear weapons are certainly dangerous and pose an existential threat to humanity. However, they possess no autonomy, and can easily be rendered inoperable by removing the core of the weapon. Often, the core of a nuclear bomb can be stored separately from the primary stage of the weapon (made of modern conventional explosives, which are also notoriously difficult to accidentally detonate), and the weapon has multiple redundant safety features built in. At no point in

history so far has a nuclear weapon been unintentionally detonated, as[19] a nuclear weapon left alone will not detonate of it's own accord. Hence, most of the concerns with the danger surrounding nuclear weapons are with the physical security of the devices themselves, as well as information security regarding the construction of such a device.

Nuclear power plants represent a similar safety hazard when containment fails and nuclear material contaminates the environment. Most famously, the Chernobyl accident in 1986: A combination of human error and design flaws caused a reactor meltdown, spreading radioactive material over the Soviet Union and much of Europe, and required the evacuation of some $50,000$ people.

Highly infectious diseases have more difficult safety concerns, as they are invisible, and can jump from human to human and spread. However, both infectious diseases and nuclear weapons are well understood, and we have mitigation strategies for both. Neither will act to circumvent the safety strategies in place, as they lack the intelligent or inclination to do so.

Many rules and regulations regarding mitigation of danger are written in blood, so the saying goes. Often, something that is not obviously dangerous becomes commonplace, and causes a lot of harm before our knowledge of it can catch up. Examples include the toxicity of asbestos, leaded gasoline, tobacco and ionising radiation, harm to the ozone layer caused by chlorofluorocarbons (CFCs), and the contribution of fossil fuels to anthropocentric climate change. Many of these remained in use for decades (or are still in use today!) and were even claimed to have health benefits before the science caught up.

However, the danger posed by AGI is far more insidious, as we are dealing with systems which may have the ability to reason about their surroundings in an intelligent way, and which may have behavioural and reasoning capabilities beyond that of humans. We cannot afford to simply learn from mistakes as we have done for asbestos and CFCs. An *unaligned*[20] superintelligent agent let loose in the world at large will quickly establish itself as the dominant force in the world, and will act only in ways that satisfy whatever it's original goal is. It would act quickly to safeguard it's existence (likely by making redundant copies of itself over the internet, much like a computer worm does) and then take actions to negate any potential threats or adversaries that prevent it from satisfying it's goals (which could include us!) Human values are complicated[21] and it's not obvious how we should encode human values. Simplistic proxies like "maximise happiness" lead to a repugnant conclusion of an agent incentivised to forcibly stimulate the pleasure centre of the human brain[22], extending their lifespan as long as possible to fulfil this end. "Remove all suffering" has an equally obvious (but undesirable) solution: exterminate all creatures that can suffer. Neither of these

[19] Though there have been close calls, in 1961 a nuclear bomber broke up mid-air over Goldsboro, North Carolina, jettisoning two nuclear bombs, both with yields of around 3-4 megatons. One of the two bombs came perilously close to detonation, arming three of it's four required mechanisms to detonate.

[20] An AGI with a goal that is not aligned with the interests of humanity.

[21] The fact that moral philosophers after hundreds of years still cannot agree on a foundation of what it means to be *morally good*, or politicians cannot agree on the best system for running a country should attest to this.

[22] Related is the *experience machine* Nozick (1974) a hypothetical machine where anyone can feel the experience of a fake reality, used as a refutation for moral hedonism: That which is morally good is the maximisation of pleasure and the minimisation of pain.

would make a good goal for an AGI to satisfy. An AGI wouldn't have "common sense", and wouldn't see these undesired outcomes as bad: It is merely doing whatever it takes to maximise it's goal. By the time we've realised the mistake and the AGI has already escaped whatever containment it was kept in, it's goal cannot be changed, as that would not maximise the current goal of the AGI.

This does not (necessarily) mean the goal of building a safe artificial general intelligence is hopeless.

Even though we may not understand the internal reasoning components of the agent, we can utilise the AIXI agent as a model of how an AGI will act. To this end we will present several AGI safety problems and how we can use our understanding of AIXI to solve or avoid them.

Although there are many questions about AGI we are able to provide clean complete (often mathematically formal) answers for, there are just as many if not (yet) more that possess no such clean answers. These, often philosophical, questions about AGI and intelligence require a different perspective than the formal mathematical view of the majority of the book Catt, Quarel et al. (2022). For instance, even if we could successfully construct an AGI to do our bidding, and that has our best interests in mind, should we? Should we ascribe moral value to such an agent, and grant it certain rights as we do with other sentient creatures?[23] Would having a "race" of intelligent machine to do our bidding not be akin to indentured servitude? It may be that in the future, AIs might campaign for rights much like minority groups did during the civil rights movement.

We don't claim to have the answers for these questions. Nonetheless, we try to provide what answers and viewpoints we can, making sure to include arguments for and against the possibility of AGI, the validity of the chosen intelligence measure, and several other poignant philosophical questions about AGI.

We hope that the book Catt, Quarel et al. (2022) can provide the reader with the required understanding to be able to further implement and develop this theory towards the ultimate goal of true AGI.

Additional Literature. The reason for the semimeasure gap in the Solomonoff prior is the fact that there exist programs which will halt and not continue to produce an output. In the UAI setting this semimeasure gap, and the machines halting, can be thought of as the death of agent, as it no longer receives percepts from the environment. It has been shown that this semimeasure death corresponds to an observation/state death of the agent (Martin et al., 2016). Additionally, it has been determined that AIXI will not perform an action that would test whether or not it is immortal, as such an action could result in death.

As we have discussed above, A(G)I Safety is an important topic of study; there has however been very little written on it. For a comprehensive study of the topic of AI Safety, the thesis Everitt (2018) serves as an excellent starting point. It starts with a review of the literature

[23] Obviously, humans don't grant all rights to other non-human animals, but laws regarding animal cruelty have been passed, so we have a preferance against the suffering of animals, to a degree.

(up to time of publication), then goes on to formally define misalignment and corruption, as well as other safety problems. The author then gives some tools which could be used to combat these. The thesis also uses the framework of causal graphs (Pearl, 1995) to explain many of the problems in AI safety. This causal formation improves the understanding of these problems. There are also several more concrete results presented including a "No Safe Lunch" theorem in the MDP setting.

The definitions of intelligence from fields including psychology, philosophy, computer science, neuroscience and more were surveyed by Legg and Hutter (2007). The authors then distill these definitions into a simple sentence which captures what most of the definitions are saying.

Intelligence measures an agent's ability to achieve goals in a wide range of environments.

The authors then go on to present a mathematical definition based on this sentence definition, known as the Legg-Hutter intelligence measure:

$$\Upsilon(\pi) := V_\xi^\pi(\epsilon)$$

where ϵ is the empty string. Discussions of other mathematically formalised definitions of intelligence such as the C-test, which is a passive version of the authors' definition, are given. Lastly, counterpoints to various criticisms of the definition are presented.

The UAI setup is a dualist setup, not physicalist setup. That is, the agent is considered separate to the environment. A physicalist intelligence measure, that is, a space-embedded and space-time embedded intelligence measure, has been derived using ideas from the Legg-Hutter intelligence and Russell's bounded optimality (Orseau and Ring, 2012).

In the UAI setup, we only consider rewards bounded between 0 and 1; this is because positive linear transformations on the reward do not change the behaviour of agents, so we can always transform any reward space to be bounded between 0 and 1. It has been shown that the Legg-Hutter intelligence can be extended to $[-1, 1]$, and interestingly that this extension is symmetric around the origin, implying that minimising this intelligence measure is essentially as difficult as maximising it (Alexander and Hutter, 2021).

One of the ways we can currently measure the intelligence of agents is to have them perform on various tasks, often games. There have been many recent successes on these game environments, including RL agents on Atari 2600 games (Mnih, Kavukcuoglu et al., 2015) and the game Go (Silver, Huang et al., 2016). Even more recently, with the assistance of techniques from game theory and counterfactual regret minimisation, AI agents have solved no-limit poker (Brown and Sandholm, 2019).

Chapter 3

On the Formalisation of Kolmogorov Complexity

This chapter has been published as

E. Catt and M. Norrish (2021). 'On the formalisation of Kolmogorov complexity'. In: *Proceedings of the 10th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pp. 291–299

Abstract

Kolmogorov complexity is an essential tool in the study of algorithmic information theory, and is used in the fields of Artificial Intelligence, cryptography, and coding theory. The formalisation of the theorems of Kolmogorov complexity is also key to proving results in the theory of Intelligent Agents, specifically the results in Universal Artificial Intelligence. In this chapter, we present a mechanisation of some of these fundamental results. In particular, building on HOL4's existing model of computability, we provide a formal definition of the complexity of a binary string, and then prove (i) that Kolmogorov complexity is uncomputable; (ii) the Kolmogorov Complexity invariance theorem; (iii) the Kraft and Kolmogorov-Kraft inequalities; and (iv) key Kolmogorov Complexity inequalities.

3.1 Introduction

Though many computational models have been formalised (including, for example, Turing machines (Xu et al., 2013), and the lambda calculus (Norrish, 2011a)), there has been comparatively little done in the area of complexity theory. Most work to date has been concerned with complexity analyses of particular algorithms and/or data structures, and most work has focused on elapsed time as the complexity measure of interest. In this chapter, we formalise a different kind of computational complexity, Kolmogorov complexity, which measures the complexity of specifying various flavours of mathematical objects (often strings). To our knowledge, we are the first to formalise both the core notion of Kolmogorov complexity, as well as the first to mechanically prove the subsequent theorems below.

Our formalisation takes place in HOL4, using its extant library of computability results (Norrish, 2011a). This library gives us access to a universal function Φ and an underlying computational model based on the λ-calculus. As a miniature programming language, the latter is useful for expressing the various algorithms that arise.

The goal of the formalisation of Kolmogorov complexity is to eventually be able to use it to formalise theorems of intelligent agents which use Kolmogorov complexity, including the (theoretically) most intelligent agent AIXI (Hutter, 2005).

3.1.1 Contributions

We carry out a novel HOL mechanisation, which:

- defines Kolmogorov complexity;
- proves the Kraft inequality;
- proves the Kolmogorov-Kraft inequality;
- proves the Kolmogorov complexity invariance theorem;
- proves that Kolmogorov complexity is not computable; and
- proves key Kolmogorov complexity inequalities.

We also believe that this is one of the first general mechanised treatment of any flavour of complexity theory. Previous work analysing time complexity has tended to focus on the verification of specific algorithms, rather than, for example, proving general results about complexity classes.

HOL results The definitions and theorems described in this chapter have been automatically rendered from the original HOL sources. The presence of $\stackrel{\text{def}}{=}$ or the turnstile ($\vdash$) indicate that this translation has been performed. The pretty-printing introduces LaTeX niceties (superscripting, symbol stacking *etc*) that are not present in the original sources. The original

sources, which add full proofs for all the theorems, are available from the `examples/-computability/kolmogorov` directory of the current HOL4 distribution (Developers, 2020).

Notation Most of our vocabulary and notation is standard mathematics (*e.g.*, set theoretic notions like union ($\cup$), the empty set $\emptyset$ and set comprehensions). We use a back-tick single-quote (f ' s) to indicate the image of set s under function f. We use the type of lists extensively, where operations include "cons" ($h :: t$ is the list consisting of element h followed by the elements of list t), length ($|xs|$), append ($xs +\!\!+ ys$), and the higher-order MAP, which maps a function across the elements of a list. Thus:

$$\mathsf{MAP}\ f\ [x;\ y;\ z] = [f\ x;\ f\ y;\ f\ z]$$

Following functional programming practice, most functions are applied in curried style, with fewer parentheses ($f\ x\ y$ rather than $f\ (x, y)$). As with SML (the Standard ML language), we use an option type (`Maybe` in Haskell) with constructors SOME and NONE. Anything with the option type is either $\mathsf{SOME}\ x$ or NONE. The THE function has defining equation $\mathsf{THE}\ (\mathsf{SOME}\ x) = x$. (Its value when applied to NONE is unspecified.)

On the natural numbers, we occasionally use a bijective pairing function: when m and n are both natural numbers, so too is $m \otimes n$.

3.1.2 Related Work

Although our work is the first formalisation of Kolmogorov complexity, there has been much work on formalisation in the related area of computability theory (*e.g.*, Carneiro (2019); Forster, Heiter et al. (2018); Norrish (2011a); Xu et al. (2013)). Work in the area of complexity theory has focused on the verification of complexities of specific algorithms and/or data structures. Recent, sophisticated examples include Nipkow and Brinkop (2019) (in Isabelle), and (Charguéraud and Pottier, 2019) (in Coq) both attacking amortised complexity arguments for complex data structures. Slightly more recently, Forster, Kunze et al. (2020) have an almost entirely mechanised proof that the weak CBV λ-calculus is a reasonable model for both time and space complexity.

Kolmogorov complexity also points in the direction of entropy, information theory and measure spaces. Thus, other related work includes formalisations of measure and probability theory by Hölzl (2013); Hölzl and Heller (2011) (Isabelle/HOL) and Mhamdi et al. (2013) (HOL Light and HOL4). Collectively, the authors formalise foundational concepts in measure theory such as the Lebesgue measure, the product measure, the extended real numbers and the Radon-Nikodym derivative. Additional results include the Chebyshev inequality, the Markov inequality and the Weak Law of Large Numbers. A formalisation of these concepts of measure theory and probability will be useful to us in a future investigation of the semi-measure properties of the universal prior, an area where probability theory and computability overlap.

3.2 Computability

The existing HOL4 library of theories about computability (described in Norrish (2011a)) builds a definition of a universal recursive function, Φ, on top of a mechanisation of the λ-calculus. The function Φ has type $\texttt{num} \to \texttt{num} \to \texttt{num option}$, so that $\Phi\ f\ x$ represents the execution of the f-th function on argument x. The option type in the result encodes the possibility that the computation may not terminate. Thus, Φ indexes into an enumeration of the recursive functions, and being recursive itself, it has an index of its own.

The enumeration used by Φ is a bijection between the natural numbers and the λ-terms.[1] It is thus possible to write a recursive function by defining a λ-term for performing that function's computation, and then applying the bijection. This generates an index that becomes Φ's first argument. This facility is very useful when we come to define computations relevant to Kolmogorov complexity.

In this domain, it is frequently useful to imagine our "machines" acting on inputs that are strings of binary digits rather than natural numbers. For example, concatenation can be much more intuitive than a shift (exponentiation) and addition. Therefore, throughout what follows, we switch between numbers and finite binary strings (lists of Boolean values). This conversion is done by mutually inverse functions: $\lceil n \rceil$ denotes the binary expansion of n, and $\lfloor s \rfloor$ is the number corresponding to a binary string s.

To avoid having to worry about leading zeroes, we use a "2-1" encoding for these functions:

$$\begin{aligned}
&\lceil n \rceil \stackrel{\text{def}}{=} \\
&\quad \texttt{if } n = 0 \texttt{ then } [\,] \\
&\quad \texttt{else if } \mathsf{EVEN}\ n \texttt{ then } \mathsf{T} :: \lceil (n-2)\ \mathsf{DIV}\ 2 \rceil \\
&\quad \texttt{else } \mathsf{F} :: \lceil (n-1)\ \mathsf{DIV}\ 2 \rceil \\
\\
&\lfloor [\,] \rfloor \stackrel{\text{def}}{=} 0 \\
&\lfloor h :: t \rfloor \stackrel{\text{def}}{=} (\texttt{if } h \texttt{ then } 2 \texttt{ else } 1) + 2 \cdot \lfloor t \rfloor
\end{aligned}$$

They are indeed inverses of each other:

$$\vdash \lceil \lfloor s \rfloor \rceil = s$$

$$\vdash \lfloor \lceil n \rceil \rfloor = n$$

[1] The Φ function is implemented as a λ-calculus function which interprets Church-encoded λ-terms (these are effectively doubly-deeply-embedded, with the interpreter being just singly-deeply embedded). This then appeals to the "beta-normal-form-of" function (with range `term option`) to return the interpreter's normal form (if any), which in turn embodies the input's normal form.

(This encoding is also used to represent numerals in HOL4, and is termed the *2-adic representation* in Smullyan (1961, p35).) We also occasionally use the length of the string corresponding to a number, writing $\ell\ n$ to mean $||\lceil n \rceil||$. Note that this function is essentially a natural number logarithm (for $0 < n$, $\ell(2^n) = n$), but it is defined for $n = 0$. For $n \leq 2$: $\ell\ 0 = 0$, $\ell\ 1 = 1$, $\ell\ 2 = 1$.

3.3 Strings and Prefix-Free Sets

For this work, the machines we work with have the type

$$\texttt{bool list} \to \texttt{bool list option}$$

Because of this, if we want our machines to be able to take multiple arguments in an input, we need a way to encode multiple arguments into a list. If the number of arguments given to the machine is fixed, then one way to do this is to use the concatenation of prefix-free encodings of the arguments. While this is not the only way to encode multiple arguments into a single string, this method also offers some important improvements on key results regarding Kolmogorov complexity, such as the inequalities.

Thus, we will be using a notion of what it is for a set of strings to be prefix-free, where $x \prec y$ means that string x is a strict prefix of string y:

$$\mathsf{prefix\text{-}free}\ A \stackrel{\text{def}}{=} \forall x\ y.\ x \in A \wedge y \in A \Rightarrow \neg(x \prec y)$$

One of the main prefix-free encodings of binary strings we will be using is called bar (Hutter, 2005).

Definition 3.1 (bar)**.** *The function* bar *takes a list x and prepends it with a list of* T*'s whose length is equal to that of x, and a single* F*.*

$$\mathsf{bar}\ x \stackrel{\text{def}}{=} \mathsf{T}^{|x|} +\!\!+ [\mathsf{F}] +\!\!+ x$$

(Note that bar *is injective, with inverse* unbar*.)*

Theorem 3.2. *This encoding is indeed prefix-free:*

$$\vdash \mathsf{prefix\text{-}free}\ (\mathsf{bar}\ `\ A)$$

with A a set of strings.

The encoding bar is called a first-order encoding. There is a second-order encoding method called dash, which when given an x returns the concatenation of bar of the length of x and x.

$$\mathsf{dash}\ x \stackrel{\text{def}}{=} \mathsf{T}^{\ell\ |x|} +\!\!+ [\mathsf{F}] +\!\!+ \lceil |x| \rceil +\!\!+ x$$

The dash encoding gives shorter encoding of strings asymptotically. However, since we only require that our encoding lead to a prefix-free code, we will be exclusively using the simpler first-order encoding bar.

We use this bar to encode multiple arguments with our pairing function, pair:

$$\vdash \mathsf{pair}\ x\ y = \mathsf{bar}\ x +\!\!+ y$$

If we want to encode more than 2 arguments, we use multiple applications of pair. For example, with arguments x, y and z, we can use pair x (pair y z). Importantly, pairing will not be prefix-free unless the second argument is itself a prefix-free encoding.

3.4 Kolmogorov Complexity

There are many definitions of what it means for something to be simple or complex. In theoretical computer science, complexity often refers to the time taken for a task to be solved; this notion is called computational complexity. Another notion is *Kolmogorov complexity* (Kolmogorov, 1963). The Kolmogorov complexity of a string/sequence/object is the length of the smallest program on a machine that can produce that object. Importantly, Kolmogorov complexity also takes the machine which runs the program as an input. Kolmogorov complexity has been used in a variety of applications, including as the basis for a similarity metric (Li, Chen et al., 2004).

Our primary reference for what follows is Hutter (2005), but we also referred to Li and Vitányi (2008).

In HOL, our definition of this foundational concept is:

Definition 3.3. *Kolmogorov Complexity*

$$\begin{aligned}&\mathsf{core\text{-}complexity}\ U\ x \stackrel{\mathsf{def}}{=}\\ &\quad \texttt{if}\ \{p \mid U\ p = \mathsf{SOME}\ x\} = \emptyset\ \texttt{then}\ \mathsf{NONE}\\ &\quad \texttt{else}\ \mathsf{SOME}\ (\mathsf{MIN\text{-}SET}\ \{|p| \mid U\ p = \mathsf{SOME}\ x\})\end{aligned}$$

One key part of the definition of Kolmogorov complexity is that if there does not exist a program which can produce the given string on the given machine, then the Kolmogorov complexity of that string on that machine is infinity. As above, we represent this by having the type of Kolmogorov complexity be

$$\begin{aligned}&(\texttt{bool list} \to \texttt{bool list option}) \to\\ &\texttt{bool list} \to \texttt{num option}\end{aligned}$$

using the option type to represent the possibility of non-termination. Thus, the NONE in `num option` represents infinity, and the option result type represents the extended naturals, $\mathbb{N} \cup \{\infty\}$.

Sometimes we are interested in the complexity of a string given some information, that is, what is the shortest program that when given some information produces the desired string. This is called conditional Kolmogorov complexity, and in HOL it is defined as:

Definition 3.4. *Conditional Kolmogorov Complexity*

$$\begin{aligned}&\textsf{cond-core-complexity}\ U\ x\ y \stackrel{\text{def}}{=}\\ &\texttt{if}\ \{p \mid U\ (\textsf{pair}\ y\ p) = \textsf{SOME}\ x\} = \emptyset\ \texttt{then}\ \textsf{NONE}\\ &\texttt{else}\ \textsf{SOME}\ (\textsf{MIN-SET}\ \{|p| \mid U\ (\textsf{pair}\ y\ p) = \textsf{SOME}\ x\})\end{aligned}$$

Here we use the pair *function to encode the extra information being given to the machine.*

Importantly, if the parameter U above is a universal machine, the corresponding Kolmogorov complexity of any string will always be finite. For the conditional Kolmogorov complexity to be finite we need to assume U is one of the types of universal machines described in Section 6 (or make some other additional assumptions). Because of this, we can define a version of core-complexity and cond-core-complexity without the option type, using THE.

Definition 3.5. *Kolmogorov Complexity (with a universal machine)*

$$K_U(x) \stackrel{\text{def}}{=} \textsf{THE}\ (\textsf{core-complexity}\ U\ x)$$

Definition 3.6. *Conditional Kolmogorov Complexity (with a universal machine)*

$$K_U(x \mid y) \stackrel{\text{def}}{=} \textsf{THE}\ (\textsf{cond-core-complexity}\ U\ x\ y)$$

We have already introduced the universal Φ, but the U above is of a different type, taking one binary string as an input, rather than two natural numbers. We link the two notions with our univM predicate.

Definition 3.7. *Universal Machines on Binary Strings*

$$\textsf{univM}\ U \stackrel{\text{def}}{=} \forall f.\ \exists g.\ \forall x.\ \lceil \Phi\ f\ x \rceil = U\ (g +\!\!+ \lceil x \rceil)$$

This definition does not capture every possible universal machine, but only those which accept a specific coding of the inputs. The particular coding will be important when we come to show some of the inequalities for Kolmogorov complexity. As desired, we are also able to

show that all machines which are universal will have a finite Kolmogorov complexity for any string with the following theorem.

Theorem 3.8.

$$\vdash \mathsf{univM}\ U\ \Rightarrow\ \{p\mid U\ p = \mathsf{SOME}\ x\}\ \neq\ \emptyset$$

3.5 Invariance Theorem and Uncomputability

Kolmogorov complexity does not only determine how complex a given string is, but it can also be used to determine how complex a machine is compared to another machine. If $K_U(x) \geq K_V(x)$ for all x, then we can say that U is a more complex machine than V. To determine what machines have this property, we have the invariance theorem which uses the univM definition from the previous sections.

Theorem 3.9 (Invariance Theorem)**.** *The invariance theorem states that the Kolmogorov complexity of a universal machine, in the sense of* univM*, is at most a constant (not in terms of x) away from any other machine.*

$$\begin{aligned}\vdash\ &\exists c.\ \forall U\ V\ i.\\ &\quad \mathsf{univM}\ U\ \wedge\ i\ \in\ \mathsf{indexes\text{-}of}\ V\ \Rightarrow\\ &\quad \forall x.\ \mathsf{core\text{-}complexity}\ U\ x\ \leq\\ &\qquad \mathsf{core\text{-}complexity}\ V\ x + \mathsf{SOME}\ (c\ U\ i)\end{aligned}$$

where k is an index of V if $V = \Phi_k$*. The use of* SOME *here indicates we are using arithmetic on the extended naturals (*$\leq$ *and addition have been lifted to this domain). Thus, the constant function asserted to exist here returns a normal natural number.*

The proof uses the fact that a universal machine can, by definition, run any other machine, and therefore can run that machine with the shortest program which computes a given x.

A key property of Kolmogorov complexity is that it is not computable. We follow the proof found in Hutter (2005). A function is defined to be computable if there exists an index such that the universal function Φ applied to that index is equal to the function.

Definition 3.10 (Computable)**.**

$$\mathsf{computable}\ f\ \stackrel{\mathsf{def}}{=}\ \exists i.\ \forall n.\ \Phi\ i\ n = \mathsf{SOME}\ (f\ n)$$

Note that f is a total function of type `num` $\rightarrow$ `num`*. By way of contrast,* Φ *is necessarily partial (its range is* `num option`*), so the use of* SOME *insists that the candidate index i is that of a (total) recursive function.*

We begin with some inequalities about Kolmogorov complexity; first an inequality about the Kolmogorov complexity of a computable function applied to some input.

Lemma 3.11.

$$\vdash \mathsf{computable}\, f \ \wedge\ \mathsf{univM}\ U \ \Rightarrow$$

$$\exists c.\ \forall m.\ K_U(\lceil f\ m\rceil) \ \leq\ \ell\ m + c$$

The proof follows from the fact that the machine in the definition of Kolmogorov complexity is universal, and so it can run the function f.

We also prove a key upper bound of Kolmogorov complexity,

Lemma 3.12.

$$\vdash \mathsf{univM}\ U \ \Rightarrow\ \exists c.\ \forall m.\ K_U(\lceil m\rceil) \ \leq\ \ell\ m + c$$

This is a consequence of the existence of a "print" machine defined as $T(x) = x$. Since there exists a machine which can do this, the Kolmogorov complexity is no greater than the sum of the length of that machine (our c above), and the length of the input to the machine, $|x|$. As expected, the lambda calculus identity function ($\lambda x.x$) is our witness in the proof of this result.

Now we can prove the uncomputability of Kolmogorov complexity.

Theorem 3.13 (Incomputability of Kolmogorov Complexity)**.**

$$\vdash \mathsf{univM}\ U \ \Rightarrow\ \neg\mathsf{computable}\ (\lambda x.\ K_U(\lceil x\rceil))$$

Proof. We assume Kolmogorov complexity is computable and show that this leads to a contradiction.

If Kolmogorov complexity is computable, then the function

$$\vdash f_U\ m = \mathsf{MIN\text{-}SET}\ \{\lfloor x\rfloor \mid m \ \leq\ K_U(x)\}$$

exists and is computable. (The f_U function can be implemented by iterating computations of KC over increasingly large strings. The first one found whose complexity is larger than argument m is the result.) Then we have

$$m \ \leq\ K_U(\lceil f_U\ m\rceil)$$

by the definition of f_U. Equally, we have from earlier that

$$\vdash \mathsf{univM}\ U \ \wedge\ \mathsf{computable}\ (\lambda x.\ K_U(\lceil x\rceil)) \ \Rightarrow$$

$$\exists c.\ \forall m.\ (\lambda x.\ K_U(\lceil x\rceil))\ (f_U\ m) \ \leq\ \ell\ m + c$$

therefore

$$\vdash \mathsf{univM}\ U\ \wedge\ \mathsf{computable}\ (\lambda x.\ K_U(\lceil x \rceil))\ \Rightarrow$$

$$\exists c.\ \forall m.\ m\ \leq\ 2 \cdot \ell\ m + c$$

Of course, as m is not bounded by $\ell\ m$ (since $\ell\ m$ is close to $\log m$), no such c exists. Thus we have a contradiction, and Kolmogorov complexity is not computable. □

3.6 Kolmogorov Complexity Inequalities

In the literature (*e.g.*, Li and Vitányi (2008)), there are two kinds of Kolmogorov complexity defined: plain Kolmogorov complexity and prefix-free Kolmogorv complexity. Prefix-free Kolmogorov complexity requires that the set of programs which halt on the given machine form a prefix-free set; plain Kolmogorov complexity does not. In this section, we will describe the inequalities of both sorts.

For Plain Kolmogorov Complexity, we use the non-prefix free version of the universal machine defined in Hutter's Theorem 2.7 (Hutter, 2005):

Definition 3.14 (The "Plain" Universal Machine)**.**

$$\mathsf{univM}_{\mathrm{pl}}\ U\ \stackrel{\mathrm{def}}{=}$$

$$(\forall i\ x\ y.\ U\ (\mathsf{pair}\ x\ (\mathsf{pair}\ i\ y)) = \lceil \Phi\ \lfloor i \rfloor\ \lfloor \mathsf{pair}\ x\ y \rfloor \rceil)\ \wedge$$

$$\forall p.\ (\forall i\ x\ y.\ p\ \neq\ \mathsf{pair}\ x\ (\mathsf{pair}\ i\ y))\ \Rightarrow\ U\ p = \mathsf{NONE}$$

Note that because pair *is injective, there is actually exactly one function satisfying* $\mathsf{univM}_{\mathrm{pl}}$*: the contrast with* univM *is that the index into the* Φ*-enumeration is given explicitly.*

Theorem 3.15. *The* $\mathsf{univM}_{\mathrm{pl}}$ *machine is a* univM *machine:*

$$\vdash \mathsf{univM}_{\mathrm{pl}}\ U\ \Rightarrow\ \mathsf{univM}\ U$$

Proof. To show univM U (see Definition 3.7) is to show that for all f, there exists a g such that, for all x

$$\lceil \Phi\ f\ x \rceil = U\ (g +\!\!+ \lceil x \rceil)$$

Take g to be pair $[\,]$ (pair $\lceil f\ \circ\ q \rceil$ $[\,]$) where q is effectively the second projection, defined such that

$$\Phi\ q\ \lfloor \mathsf{pair}\ x\ y \rfloor = \mathsf{SOME}\ \lfloor y \rfloor$$

Note we are using function composition with natural number arguments here: this operation returns an index for the function that is the composition of the functions indexed by the two arguments. Our choice of g suits the way in which our particular U is required to process its

arguments by the definition of $\mathsf{univM}_{\mathrm{pl}}$. And so:

$$\begin{aligned} U\ (g +\!\!+ x) &= U\ (\mathsf{pair}\ [\,]\ (\mathsf{pair}\ \lceil f \circ q \rceil\ [\,]) +\!\!+ x) \\ &= U\ (\mathsf{pair}\ [\,]\ (\mathsf{pair}\ \lceil f \circ q \rceil\ x)) \\ &= \lceil \Phi\ (f \circ q)\ \lfloor \mathsf{pair}\ [\,]\ x \rfloor \rceil \\ &= \lceil \Phi\, f\ (\mathsf{THE}\ (\Phi\, q\ \lfloor \mathsf{pair}\ [\,]\ x \rfloor)) \rceil \\ &= \lceil \Phi\, f\ x \rceil \end{aligned}$$

□

Definition 3.16 (The Prefix-Free Universal Machine)**.** *For prefix-free Kolmogorov complexity, we define a different notion of universal machine (also from Hutter (2005, Theorem 2.7)):*

$$\begin{aligned} &\mathsf{univM}_{\mathrm{pf}}\ U \stackrel{\mathrm{def}}{=} \\ &(\forall i\ x\ y.\ U\ (\mathsf{pair}\ x\ (\mathsf{pair}\ i\ y)) = \\ &\qquad \lceil \Phi\ \mathsf{Upfi}\ (\lfloor i \rfloor \otimes \lfloor \mathsf{pair}\ x\ y \rfloor) \rceil) \wedge \\ &\forall m.\ (\forall i\ x\ y.\ m \neq \mathsf{pair}\ x\ (\mathsf{pair}\ i\ y)) \Rightarrow U\ m = \mathsf{NONE} \end{aligned}$$

Note how the only difference between this definition and Definition 3.14 is the way in which Φ is called.

Instead of using using the universal function Φ, prefix-free Kolmogorov complexity uses a prefix-free universal machine, which has index Upfi. The construction of this machine and the proof of its correctness are both from Downey and Hirschfeldt (2010).

Definition 3.17 (Prefix-free Indices)**.** *Call a machine-index i prefix-free if the domain of Φ_i is prefix-free:*

$$\mathsf{pfi}\ i \stackrel{\mathrm{def}}{=} \mathsf{prefix\text{-}free}\ \{\lceil x \rceil \mid \Phi\ i\ x \neq \mathsf{NONE}\}$$

Note that we can create prefix-free indices to emulate any Φ_j by constructing j' which takes inputs $\mathsf{bar}\ x$ *to $\Phi_j(x)$ and loops otherwise.*

Theorem 3.18 (Upfi Properties)**.** *We have the following results about Φ Upfi, which we call the "Upfi-machine":*

- *The Upfi-machine is indeed universal; and*
- *Given a machine M to work with, the Upfi-machine's emulation of it creates a machine with a prefix-free domain*

In HOL:

$$\vdash \mathsf{pfi}\ M\ \Rightarrow\ \Phi\ \mathsf{Upfi}\ (M\ \otimes\ i) = \Phi\ M\ i$$

$$\vdash \forall M.\ \mathsf{prefix\text{-}free}\ \{\lceil i \rceil\ \mid\ \Phi\ \mathsf{Upfi}\ (M\ \otimes\ i)\ \neq\ \mathsf{NONE}\}$$

(Recall: $M\ \otimes\ i$ is the application of our bijective pairing function over the natural numbers.)

Theorem 3.19 (The Prefix-Free Universal Machine is Prefix-Free)**.** *The final theorem above, coupled with the* pair *function, ensure that the inputs for which the* $\mathsf{univM}_{\mathrm{pf}}$ *machine halts form a prefix-free set:*

$$\vdash \mathsf{univM}_{\mathrm{pf}}\ U\ \Rightarrow\ \mathsf{prefix\text{-}free}\ \{p\ \mid\ U\ p\ \neq\ \mathsf{NONE}\}$$

Many of the following results are roughly of the shape

$$\textit{complexity-measure}(x) \leq \textit{complexity-measure}(y) + c$$

where c is a constant independent of arguments x and y. The usual technique for proving these inequalities is to show how the computation of x can be achieved given a value y. The minimal machine that computes y can be composed with a machine that transforms such inputs into x. The latter is of constant size, and so too is the composition of the two machines. The minimal machine computing x must therefore be of size no larger than that composition. Thus, our proofs build recursive functions which are equivalent to operations over strings such as bar, pair, concatenation, un-pairing, *etc*.

The names for the following results are taken from Hutter (2005).

Theorem 3.20 (Extra Information 1)**.** *Providing extra information* (i.e., *we are using conditional Kolmogorov complexity), does not increase the Kolmogorov complexity by more than a constant:*

$$\vdash \mathsf{univM}_{\mathrm{pl}}\ U\ \Rightarrow\ \exists c.\ \forall x\ y.\ K_U(x \mid y)\ \leq\ K_U(x) + c$$

$$\vdash \mathsf{univM}_{\mathrm{pf}}\ U\ \Rightarrow\ \exists c.\ \forall x\ y.\ K_U(x \mid y)\ \leq\ K_U(x) + c$$

Because conditional complexity provides extra information, it is enough to be able to write a machine that uniformly ignores that additional information.

Theorem 3.21 (Extra Information 2)**.** *Similarly, the complexity of computing x is less than the complexity of computing x paired with any other string y:*

$$\vdash \mathsf{univM}_{\mathrm{pl}}\ U\ \Rightarrow\ \exists c.\ \forall x\ y.\ K_U(x)\ \leq\ K_U(\mathsf{pair}\ x\ y) + c$$

$$\vdash \mathsf{univM}_{\mathrm{pf}}\ U\ \Rightarrow\ \exists c.\ \forall x\ y.\ K_U(x)\ \leq\ K_U(\mathsf{pair}\ x\ y) + c$$

Again, the proof is to show the existence of a machine of constant length which returns the first element of a pair.

Theorem 3.22 (Subadditivity 1)**.** *The complexity of a concatenation of two strings is no larger than the complexity of the pairing of those two strings:*

$$\vdash \mathsf{univM}_{\mathrm{pl}}\ U\ \Rightarrow$$

$$\exists c.\ \forall x\ y.\ K_U(x +\!\!+ y)\ \leq\ K_U(\mathsf{pair}\ x\ y) + c$$

$$\vdash \mathsf{univM}_{\mathrm{pf}}\ U\ \Rightarrow$$

$$\exists c.\ \forall x\ y.\ K_U(x +\!\!+ y)\ \leq\ K_U(\mathsf{pair}\ x\ y) + c$$

The proof is to demonstrate the existence of a machine of constant size which un-pairs and then concatenates the input. This is done through composition of the first-element-of-a-pair machine, the second-element-of-a-pair machine, and the concatenation machine.

Theorem 3.23 (Subadditivity 2)**.**

$$\vdash \mathsf{univM}_{\mathrm{pl}}\ U\ \Rightarrow$$

$$\exists c.\ \forall x\ y.\ K_U(x) + K_U(y \mid x)\ \leq\ K_U(x) + K_U(y) + c$$

$$\vdash \mathsf{univM}_{\mathrm{pf}}\ U\ \Rightarrow$$

$$\exists c.\ \forall x\ y.\ K_U(x) + K_U(y \mid x)\ \leq\ K_U(x) + K_U(y) + c$$

This result is a consequence of Extra Information part 1.

Theorem 3.24 (Symmetry of Information 1)**.** *The complexity of the pair of x and y (in that order), is no larger than the complexity of the pair of y and x (in that order).*

$$\vdash \mathsf{univM}_{\mathrm{pl}}\ U\ \Rightarrow$$

$$\exists c.\ \forall x\ y.\ K_U(\mathsf{pair}\ x\ y)\ \leq\ K_U(\mathsf{pair}\ y\ x) + c$$

$$\vdash \mathsf{univM}_{\mathrm{pf}}\ U\ \Rightarrow$$

$$\exists c.\ \forall x\ y.\ K_U(\mathsf{pair}\ x\ y)\ \leq\ K_U(\mathsf{pair}\ y\ x) + c$$

The proof is to construct the machine that pulls apart a pair and re-assembles one with arguments switched.

Theorem 3.25 (Extra Information 3)**.** *When working with conditional complexity, extra information can be ignored just as it was with $K_U(x)$ (as in Theorem 3.21).*

$$\begin{aligned}&\vdash \mathsf{univM}_{\mathrm{pl}}\ U\ \Rightarrow\\ &\quad \exists c.\ \forall x\ y\ z.\ K_U(x \mid \mathsf{pair}\ y\ z)\ \leq\ K_U(x \mid y) + c\end{aligned}$$

Theorem 3.26 (Symmetry of Information 2)**.** *The plain complexity of the pair of x and y is bounded by the sum of the plain complexity of y, and the plain conditional complexity of x given y and y's complexity, along with a non-constant term equivalent to twice the length of y.*

$$\begin{aligned}&\vdash \mathsf{univM}_{\mathrm{pl}}\ U\ \Rightarrow\\ &\quad \exists c.\ \forall x\ y.\\ &\qquad K_U(\mathsf{pair}\ x\ y)\ \leq\\ &\qquad K_U(x \mid \mathsf{pair}\ y\ \lceil K_U(y) \rceil) + K_U(y) + 2 \cdot |y| + c\end{aligned}$$

This result is rather painful to state, is not easy to prove, and has an ugly non-constant additive factor. To have only an additive constant requires prefix-free Kolmogorov complexity; without being prefix-free, the additional log-term is inescapable. (The prefix-free version of the theorem is proved in Gács (1974).) The trouble here comes from having two elements on the right side of the inequality while only having one on the left, so an encoding combining the two is required. The $K_U(y)$ term in the conditional Kolmogorov complexity is used to determine the size of the second program so an encoding of the two can be separated.

Theorem 3.27 (Subadditivity 3)**.** *Similarly, we can bound the complexity of a pair by the sum of the complexity of the first component, and the conditional complexity of the second given the first. Again we end with an unfortunate non-constant additive component.*

$$\begin{aligned}&\vdash \mathsf{univM}_{\mathrm{pl}}\ U\ \Rightarrow\\ &\quad \exists c.\ \forall x\ y.\\ &\qquad K_U(\mathsf{pair}\ x\ y)\ \leq\ K_U(x) + K_U(y \mid x) + 2 \cdot |x| + c\end{aligned}$$

The result follows immediately from the two previous theorems.

As *per* Hutter (2005), the proofs of the stronger symmetry results associated with prefix-free complexity are non-trivial, and we leave them for future work.

3.7 Kraft and Kraft-Kolmogorov Inequalities

When dealing with sets of prefix-free strings, one key result is the Kraft inequality.

The first half of the result provides "sizes" for sets of prefix-free strings. In particular, if the set $\mathcal{P}(\subset \{0,1\}^*)$ is prefix-free, then

$$\sum_{\mathcal{P}} 2^{-|p|} \leq 1$$

where $|x|$ is the length of the string x.

Example 3.28. *For example, given the prefix-free set* $\mathcal{P} = \{1, 01, 001, 0001, 00001\}$ *we can compute*

$$\sum_{p \in \mathcal{P}} 2^{-p} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} = \frac{31}{32} \leq 1$$

To prove this, we map each element x to a subset of $[0,1]$ starting at $0.x$ of length $2^{-|x|}$. We then use the prefix-free fact to show that none of these sets intersect; therefore the sum of the lengths is bounded by the length of $[0,1)$, which is 1.

We map binary strings to intervals with the interval-bl function, which is given a binary list s, and returns the interval in the form of a starting point and its length. The start of the interval is the list viewed as a binary fraction, giving a value in $[0,1)$. For example, the list $[0;\ 1;\ 1;\ 1;\ 0]$ becomes 0.01110_2, *i.e.*, $\frac{7}{16}$.

$$\text{interval-bl } \ell \stackrel{\text{def}}{=} (0.\ \ell, 2^{-|\ell|})$$

where the $0.\ \ell$ notation above is the expression converting ℓ to a natural number and dividing that number by $2^{-|\ell|}$. Because of the references to $|\ell|$, here the leading zeroes in ℓ *are* significant, and we use the traditional 0-1 binary number encoding.

We define what disjointness means for two intervals (treating them as open at the right) with the disjoint-interval function:

$$\text{disjoint-interval } (m,i)\ (n,j) \stackrel{\text{def}}{=}$$

$$\text{DISJOINT } \{r \mid m \leq r \wedge r < m+i\}$$

$$\{r \mid n \leq r \wedge r < n+j\}$$

Then we show that if a set is prefix-free, the intervals will be disjoint:

$$\begin{aligned}
&\vdash \textsf{prefix-free } P \iff \\
&\quad (\texttt{let} \\
&\qquad is = \textsf{interval-bl } `\ P \\
&\quad \texttt{in} \\
&\qquad \forall i_1\ i_2.\ i_1 \in is \wedge i_2 \in is \wedge i_1 \neq i_2 \Rightarrow \\
&\qquad\quad \textsf{disjoint-interval } i_1\ i_2)
\end{aligned}$$

To simplify the handling of infinite prefix-free sets (calculating sums over such sets requires a treatment of infinite limits), we can consider subsets of strings up to a particular length. (There are only finitely many bit-strings of length $\leq n$.) The size of a subset of P up to size n is given by $\textsf{bls-size } P\ n$, with definition:

$$\textsf{bls-size } P\ n \stackrel{\text{def}}{=} \sum\ (\lambda s.\ 2^{-|s|})\ \{x \mid x \in P \wedge |x| < n\}$$

The universal quantifier in our final result (below) allows us to suggest the infinite limit in an elementary way:

Theorem 3.29 (Kraft's Inequality (direction 1))**.**

$$\vdash \textsf{prefix-free } P \Rightarrow \forall n.\ \textsf{bls-size } P\ n \leq 1$$

There is a converse result to the above that is also of interest.

Theorem 3.30 (Kraft's Inequality (direction 2))**.** *Given a sequence ws of desired code lengths (widths), if those lengths do not exceed the size bound, there is a prefix-free code that maps codes of the required lengths to the corresponding indices of the sequence. In HOL:*

$$\begin{aligned}
&\vdash \Sigma_l\ (\textsf{MAP } (\lambda n.\ 2^{-n})\ ws) \leq 1 \Rightarrow \\
&\quad \exists P\ b. \\
&\qquad \textsf{prefix-free } P \wedge \textsf{BIJ } b\ (\textsf{count } |ws|)\ P \wedge \\
&\qquad \forall i.\ i < |ws| \Rightarrow |b\ i| = \textsf{EL } i\ ws
\end{aligned}$$

where Σ_l calculates the sum of a list of real values and count *n is the set of natural numbers less than n ($\{0 \ldots n-1\}$). Thus b is a bijection from ws's valid indices into the code elements P. The last*

conjunct states that each code does indeed have the required length.

The proof of this result builds a binary tree with values at the leaves (only) and with keys that are the sequences of binary choices that take one from the root to a leaf. These "code-trees" are defined as an algebraic type with three constructors:

$$\begin{aligned}&\alpha\ \texttt{codetree} =\\ &\quad \mathsf{Empty}\\ &\quad \mid \mathsf{Item}\ \alpha\\ &\quad \mid \mathsf{FullNode}\ (\alpha\ \texttt{codetree})\ (\alpha\ \texttt{codetree})\end{aligned}$$

(Though polymorphic, our result above only requires natural number values; α is always instantiated with `num`.)

Given a sequence of desired code-lengths, a tree can be built iteratively through repeated applications of the

$$\mathsf{insert} : \texttt{num} \to \alpha \to \alpha\ \texttt{codetree} \to \alpha\ \texttt{codetree}\ \texttt{option}$$

function, where the first argument is the desired code-length and the option type in the result allows for the possibility that the tree is too full to accommodate the requested length (the smaller the number, the more difficult it is to satisfy the request).

Theorem 3.31 (Free space guarantees insert)**.** *Because* insert *preferentially inserts to the left (maintaining the* packed *predicate), it is sure to find a working path if one is possible*

$$\begin{aligned}&\vdash \mathsf{packed}\ ct_0\ \wedge\ 2^{-n}\ \leq\ \mathsf{free\text{-}space}\ ct_0\ \Rightarrow\\ &\quad \exists ct.\ \mathsf{insert}\ n\ v\ ct_0 = \mathsf{SOME}\ ct\ \wedge\\ &\qquad \mathsf{free\text{-}space}\ ct = \mathsf{free\text{-}space}\ ct_0 - 2^{-n}\end{aligned}$$

The packed *predicate guarantees that space isn't wasted by having multiple open paths of the same length. The* free-space *function calculates a code-tree's free space (a real value in* $[0, 1]$*).*

Example 3.32. *If we wish to code the 26 letters of the alphabet with codes of length 4 for the 5 vowels, and codes of length 5 for the remaining letters, we have sufficient space (*$\frac{5}{2^4} + \frac{21}{2^5} < 1$*).*

If the first 5 calls to insert *are made for the vowels at length 4, followed by 21 calls for the consonants at length 5, the resulting code is:*

A	*0000*	*H*	*01111*	*O*	*0011*	*U*	*0100*
B	*01010*	*I*	*0010*	*P*	*10101*	*V*	*11010*
C	*01011*	*J*	*10000*	*Q*	*10110*	*W*	*11011*
D	*01100*	*K*	*10001*	*R*	*10111*	*X*	*11100*
E	*0001*	*L*	*10010*	*S*	*11000*	*Y*	*11101*
F	*01101*	*M*	*10011*	*T*	*11001*	*Z*	*11110*
G	*01110*	*N*	*10100*				

As our earlier results about decoding and encoding pairs in support of complexity results may have hinted, a sequence of bits using a prefix-free encoding such as this one need not separately indicate where code-points begin and end; any bit-sequence (of correct length) can be decoded into just one sequence of letters.

Two related areas where Kolmogorov complexity is used are in sequence prediction and the study of intelligent agents. In both cases, Kolmogorov complexity is a key component of the prior for Bayesian sequence prediction (Hutter, 2007; Rathmanner and Hutter, 2011) and for Bayesian agents (Hutter, 2005). Specifically, using $2^{-K_U(x)}$, which is known as the universal prior (Hutter, 2007; Rathmanner and Hutter, 2011).

The philosophical motivation behind this prior comes from two principles: the well-known Occam's Razor, and the less well-known Epicurus' principle of multiple explanations. Occam's Razor states that "Entities should not be multiplied beyond necessity", and Epicurus' principle of multiple explanation states that we should keep all theories which are consistent with the observations. Epicurus' principle is realised with the universal prior always being positive,

$$\vdash 0 < 2^{-K_U(x)}$$

and Occam's Razor is satisfied as the simplest solution will have a small Kolmogorov complexity, and hence $2^{-K_U(x)}$ is larger.

To be able to use the Kolmogorov prior $2^{-K_U(x)}$, we need to show that it satisfies some important properties, namely that the sum of the prior over every input is bounded above by 1. This bound is called the Kolmogorov-Kraft Inequality.

Theorem 3.33 (Kolmogorov-Kraft Inequality)**.**

$$\vdash \mathsf{univM}_{\mathrm{pf}}\ U\ \Rightarrow$$

$$\mathsf{bls\text{-}size}\ \{x\ |\ (\exists y.\ U\ x = \mathsf{SOME}\ y)\}\ n\ \leq\ 1$$

This comes from two results: first the Kraft inequality (Theorem 3.29) and secondly, the fact that our defined Kolmogorov complexity is a prefix code.

We have shown the Kraft inequality is true for any prefix code, as well as shown that if our machine is $\mathsf{univM}_{\mathrm{pf}}$, then Kolmogorov complexity forms a prefix code.

3.8 Reflections on Mechanisation

Computational Model Our use of the λ-calculus as our underlying computational model is somewhat unusual: much of the literature we worked from provides intuitions, and some proofs, in terms of Turing Machines and their tapes. Using the λ-calculus is also probably strictly unnecessary: one could work purely with the notions provided by the theory of the partial recursive functions. This would allow the construction of the functions necessary to the various results by using composition, projection, primitive recursion and minimisation. In places, our work does use this approach, but it is often tedious to have to explicitly encode everything as numbers.[2] Nor is it particularly straightforward to even write functions down.

The λ-calculus provides much nicer facilities for writing functions: one can use function parameters instead of having to write in a point-free style, and rich data types are easy to generate on-the-fly, using Church encodings. Other, earlier work by Norrish (2006, 2011b) on computability and the λ-calculus means that working on this foundation in HOL4 is generally quite pleasant. For example: though the calculus is deeply embedded, it is possible to use the HOL4 simplifier to symbolically evaluate λ-terms, deriving β-reduction equivalences, and making proofs of programs much easier to achieve than with usual operational semantics. With the recent experience of Forster, Kunze et al. (2020) suggesting that the λ-calculus is a good model for traditional time and space complexity, we feel that eschewing Turing Machines is becoming ever more justifiable.

Scale of Work For the most part, our mechanisation successfully reconstructs the proofs from the literature. The inevitable question is "at what cost?" Table 3.1 presents the numbers of lines of HOL4 script/code for our various results. Perhaps inevitably, background results about "trivialities" are a not insignificant proportion of the whole. These are never difficult, or particularly interesting, but still require a time investment to even state.

The uncomputability result works out quite smoothly. Indeed, roughly a third of the script in terms of lines of proof is given over to results about the way in which ℓ resembles logarithm (*e.g.*, $\vdash \ell\ (x \cdot y) \ \leq \ \ell\ x + \ell\ y + 1$), meaning that the core argument is rendered quite elegantly. The Kraft inequality results are also both fairly straightforward. Reasoning about the code-trees for the second direction does inflate the line count a little, but HOL4 and other ITP systems are well-suited to proving properties of inductive types such as these.

Finally, the Kolmogorov inequalities are the most involved. They rely on constructing various functions which act on boolean lists, and showing that these functions are recursive

[2] In Lean's richer type system, Carneiro (2019) is able to hide the encoding and decoding functions, and define type classes of "computable" values.

Table 3.1: Lines of source code for various parts of our mechanisation.

Theorems	Lines of Code
Boolean list and natural number background	1250
Kraft Inequality (both directions)	2031
$K_U(x)$ definitions and invariance theorem	254
Kolmogorov Incomputability	528
Kolmogorov Inequalities (Plain)	822
Kolmogorov Inequalities (Prefix-free)	929

by writing the primitive recursive or lambda calculus versions of those functions. An additional difficulty with the Kolmgorov inequalities arises with inequalities that rely on constructing functions which have to take two arguments in one list, such as Theorem 3.26 and Theorem 3.27.

3.9 Conclusion

While this chapter represents the start of formalisation of Kolmogorov complexity and Algorithmic information theory, there are still many theorems to be mechanised about Kolmogorov complexity. These include some additional inequalities about Kolmogorov complexity (particulally prefix-free Kolmogorov complexity), the relationship between Kolmogorov complexity and Shannon entropy (Leung-Yan-Cheong and Cover, 1978), Solomonoff's distribution and its relationship to Kolmogorov complexity, and the fact that Kolmogorov complexity is co-enumerable (giving it its position in the arithmetic hierarchy). Additionally we could generalise our pair function to be an arbitrary prefix-free pairing function which is computable and has a computable inverse (this could be a higher order coding, beyond bar and dash). Some of these areas would require substantial additional formalisation, such as the formalisation of the arithmetic hierarchy.

We would like to prove that the converse Kraft inequality result (Theorem 3.30) has a computable counterpart: if a computable function enumerates a sequence of code-lengths, we can construct a machine that enumerates the corresponding codes. In effect, this is making the argument that the insert function on code-trees is computable. With this result in hand, we would be in a position to prove the Minimum Description Length bound theorem (Hutter, 2005).

We would additionally like to use these results to prove theorems about intelligent agents, as well as related concepts.

Chapter 4

Reinforcement Learning with Information-Theoretic Actuation

This chapter has been published as

E. Catt, M. Hutter et al. (2022). 'Reinforcement Learning with Information-Theoretic Actuation'. In: *Proc. 15th Conf. on Artificial General Intelligence (AGI'22)*. Vol. 13539. LNCS. Seattle, USA: Springer

Abstract

Reinforcement Learning formalises an embodied agent's interaction with the environment through observations, rewards and actions. But where do the actions come from? Actions are often considered to represent something external, such as the movement of a limb, a chess piece, or more generally, the output of an actuator. In this work we explore and formalize a contrasting view, namely that actions are best thought of as the output of a sequence of internal choices with respect to an action model. This view is particularly well-suited for leveraging the recent advances in large sequence models as prior knowledge for multi-task reinforcement learning problems. Our main contribution in this work is to show how to augment the standard MDP formalism with a sequential notion of internal action using information-theoretic techniques, and that this leads to self-consistent definitions of both internal and external action value functions.

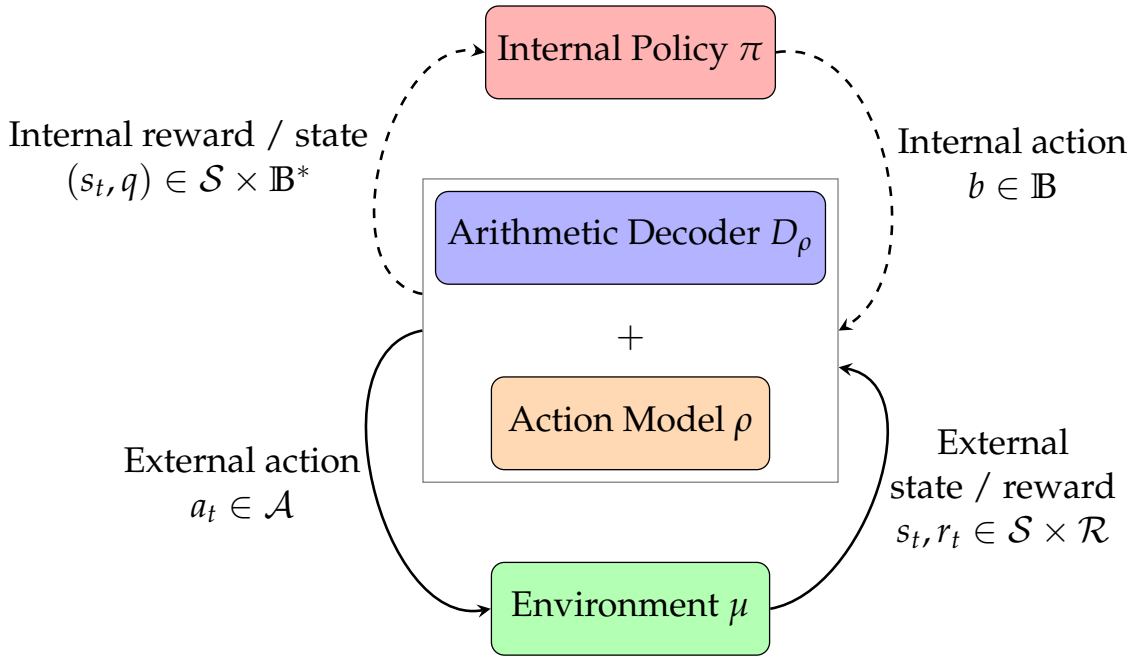

Figure 4.1: Agent-environment loop with internal actions.

4.1 Introduction

It is hard to speak of embodied agents these days without mentioning or appealing to some notion of Reinforcement Learning. This particular mathematical formalism has been so successful of late that the validity of its various modelling assumptions rarely gets called into question. Yet recently we have seen a step-change in the capabilities of generative modelling, with the most striking example being in multi-modal language applications; the acquisition of gigantic multi-task datasets via internet scraping and scalable approaches to training has led to a renewed excitement for building next generation question-answering systems, chat bots, productivity tools, sentiment analysis, and in some circles, has even produced a newfound sense of optimism that the original goals of Artificial Intelligence may well be obtainable within our lifetimes.

Yet what does this mean for Reinforcement Learning? While its success in restricted domains is no longer in doubt, questions remain about its long-term viability as a foundational paradigm for Artificial Intelligence. For example, effective exploration, even in restricted settings such as finite MDPs, is problematic in large unstructured state spaces, with various lower bounds demonstrating polynomial dependence on the size of the state space, e.g. Strehl et al. (2009). While there are some noteworthy recent examples of hard exploration problems being overcome by clever heuristics (Ecoffet et al., 2021), the situation in general looks challenging, if not dire. On the other hand, recent advances in sequence modelling combined with the acquisition of gigantic datasets via internet scraping has led to a seeming step-change (Brown, Mann et al., 2020) in the ability of various types of probabilistic models to generate plausible continuations. Is there a way to leverage this, while keeping the basic reinforcement learning formalism and derived notions such as value functions, policies, return, etc intact?

Our proposal argues for rethinking the fundamental notion of action in reinforcement learning. Actions are often considered to represent something external, such as the movement of a limb, a chess piece, or more generally, the output of an actuator. In this work however, we develop a generic notion of *internal* action, which is implied by a choice of action model ρ. The key technical insight we leverage is the well-known duality between optimal lossless coding strategies and probabilities from information theory. At a high level, instead of an agent directly picking an action from the action space $\mathcal{A}$, instead it will pick a sequence of internal actions from an internal action set $\mathbb{B}$ which will decode to an external action from $\mathcal{A}$. Figure 4.1 depicts this interaction graphically.

So what do we gain by introducing this particular layer of indirection in the agent's choice of action? Breaking up an action into a series of internal actions seems like a reasonable approach to dealing with large action spaces, and indeed has been used in other planning settings, but it immediately throws up a number of questions. How do we decompose an arbitrary action space? Is there a universal, or in some sense optimal decomposition? When should the agent stop generating internal actions and communicate an external action to the environment? Does this even make sense in a reinforcement learning setting? How do we leverage prior knowledge in the form of a default policy? Are there ramifications for multi-task RL? Can we efficiently compute or sample good actions? This chapter will argue that our particular information-theoretic decomposition using an arithmetic decoder coupled with a coding distribution implied by a choice of action model naturally addresses all these questions, and opens up the possibly of leveraging recent advances in meta-learning and large-scale language/sequence models to deal with large problems using existing RL techniques.

Content. The chapter is structured as follows: Section 4.2 reviews some background material and establishes some notation; Section 4.3 introduces internal actions, with Section 4.4 formally establishing the connection between internal/external agents and environments; Section 4.5 shows how the internal action framework naturally accommodates multi-task reinforcement learning settings with different action spaces. We conclude with an extended discussion in Section 4.6 and cover related and future work in Sections 4.7 and 4.8.

4.2 Preliminaries

We now briefly review the necessary background material required to describe our internal action agent-environment interaction loop.

Sequential Prediction. A finite alphabet $\mathcal{X}$ is a set of symbols. A string of symbols $x_1 x_2 \dots x_n \in \mathcal{X}^n$ of length n is denoted by $x_{1:n}$. The prefix $x_{1:j}$ of $x_{1:n}$, $j \leq n$, is denoted by $x_{\leq j}$ or $x_{<j+1}$. The empty string is denoted by ϵ. The set of strings whose symbols come from the alphabet $\mathcal{X}$ with length at most n is defined by $\mathcal{X}^{\leq n} := \{\epsilon\} \cup \bigcup_{i=1}^{n} \mathcal{X}^i$. The set of strings of symbols from alphabet $\mathcal{X}$ with finite length is denoted by $\mathcal{X}^* := \{\epsilon\} \cup \bigcup_{i=1}^{\infty} \mathcal{X}^i$.

The concatenation of two strings x and y is denoted by xy. The length of a string x will be denoted by $|x|$. We will use $y \in x$ to denote that the symbol y is in the string x.

A (coding) distribution ρ is a sequence of probability mass functions $\rho_n : \mathcal{X}^n \to [0,1]$, which for all $n \in \mathbb{N}$ satisfy the constraint that $\rho_n(x_{1:n}) = \sum_{y \in \mathcal{X}} \rho_{n+1}(x_{1:n}y)$ for all $x_{1:n} \in \mathcal{X}^n$, with the base case $\rho_0(\epsilon) := 1$. From here onwards, whenever the meaning is clear from the argument to ρ, the subscript on ρ will be dropped. Under this definition, the conditional probability of a symbol x_n given previous data $x_{<n}$ is defined as $\rho(x_n|x_{<n}) := \rho(x_{1:n})/\rho(x_{<n})$ provided $\rho(x_{<n}) > 0$, with the familiar chain rules $\rho(x_{1:n}) = \prod_{i=1}^{n} \rho(x_i|x_{<i})$ and $\rho(x_{j:k} \mid x_{<j}) = \prod_{i=j}^{k} \rho(x_i|x_{<i})$ now following. We will use $\Delta(\mathcal{X})$ to denote the space of probability distributions over $\mathcal{X}$.

Arithmetic Encoding / Decoding. A fundamental technique known as *arithmetic encoding* (Rissanen and Langdon, 1979; Witten et al., 1987) makes explicit the connection between coding distributions and source codes. Binary arithmetic encoding is a general purpose parameterized technique that takes in a distribution ρ (known as a coding distribution) and some data $x_{1:n} \in \mathcal{X}^n$, and produces a uniquely decodable binary codeword $C_\rho(x_{1:n}) \in \{0,1\}^*$, whose length is essentially $\lceil -\log_2 \rho(x_{1:n}) \rceil$, which is optimal in terms of expected length if the data is sampled from ρ. In essence, shorter binary codewords are assigned to data which has a higher chance of occurring under ρ, and longer binary codewords are assigned to the less probable data items. Arithmetic decoding is the reverse of this procedure; it takes a coding distribution ρ, a binary code word $y_{1:k} = C_\rho(x_{1:n})$, and returns the original data $D_\rho(y_{1:k}) = x_{1:n}$. We will also use the shorthand notation $D_\rho(y_{1:k} \mid s) := D_{\rho(\cdot|s)}(y_{1:k})$ to denote decoding with respect to a coding distribution ρ conditioned on the string s. We refer the reader to the standard text of Cover (1999) for further information.

Markov Decision Processes. A Markov Decision Process (MDP) is a type of probabilistic model widely used within reinforcement learning (Sutton and Barto, 2018; Szepesvári, 2010) and control (Bertsekas and Tsitsiklis, 1996). In this work, we limit our attention to finite horizon, time-homogeneous MDPs whose action and state spaces are finite. Formally, an MDP is a quadruplet $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mu)$, where $\mathcal{S}$ is a finite, non-empty set of states, $\mathcal{A}$ is a finite, non-empty set of actions, $\mathcal{R} \subset \mathbb{R}$ is the reward space, and μ is the transition probability kernel that assigns to each state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$ a probability measure $\mu(\cdot \mid s,a)$ over $\mathcal{S} \times \mathcal{R}$. $\mathcal{S}$ and $\mathcal{A}$ are known as the *state space* and *action space* respectively. The transition probability kernel gives rise to the *state transition kernel* $\mathcal{P}(s'|s,a) := \mu(\{s'\} \times \mathcal{R} \mid s,a)$, which gives the probability of transitioning from state s to state s' if action a is taken in s.

An agent's behavior is determined by a *policy* that defines, for each state $s \in \mathcal{S}$ and time $t \in \mathbb{N}$, a probability measure over $\mathcal{A}$ denoted by $\pi_t(\cdot \mid s)$. A *stationary policy* is a policy which is independent of time, which we will denote by $\pi(\cdot \mid s)$ where appropriate. At each time t, the agent communicates an action $A_t \sim \pi_t(\cdot \mid S_{t-1})$ to the system in state $S_{t-1} \in \mathcal{S}$. The system then responds with a state-reward pair $(S_t, R_t) \sim \mu(\cdot \mid S_{t-1}, A_t)$. Here we will assume that each reward is bounded between $[r_{\min}, r_{\max}] \subset \mathbb{R}$ and that the system starts in a state

s_0 and executes for an infinite number of steps. Thus the execution of the system can be described by a sequence of random variables $S_0, A_1, S_1, R_1, A_2, S_2, R_2, ...$

The finite m-horizon *return* from time t is defined as $Z_t := \sum_{i=t}^{t+m-1} R_i$. The expected m-horizon return from time t, also known as the *value function*, is denoted by $V_\mu^\pi(s_t) := \mathbb{E}[Z_{t+1} \mid S_t = s_t]$. The return space $\mathcal{Z}$ is the set of all possible returns. The *action-value function* is defined by $Q_\mu^\pi(s_t, a_{t+1}) := \mathbb{E}[Z_{t+1} \mid S_t = s_t, A_{t+1} = a_{t+1}]$. An *optimal policy*, denoted by π_μ^*, is a policy that maximizes the expected return $\mathbb{E}\left[Z_{t+1} \mid S_t\right]$ for all t.

4.3 Information-Theoretic Actuation – Internal Actions

We now describe in detail how to combine the aforementioned building blocks into the internal reinforcement learning framework described in Figure 4.1, and discuss its ramifications. Compared with the standard agent-environment loop, there are two additional components with this setup: a choice of action model ρ, and an associated arithmetic decoder D_ρ that uses ρ as a coding distribution. The internal action space $\mathbb{B}$ is defined by the associated decoding alphabet used by D_ρ; for example, using a binary arithmetic decoder would lead to an internal action space of $\mathbb{B} = \{0, 1\}$. For pedagogical purposes, we will restrict our attention to this case in the rest of the chapter, but remark that any finite decoding alphabet can in principle be used with our construction.

We first introduce our notion of internal action. At a high-level, one should think of a single internal action as a bit-commitment towards a particular choice of external action, with particular sequences of these corresponding to external actions. In a sense, internal actions correspond to a period of private deliberation by the agent, which upon conclusion produces a string describing the desired actuation in compressed form; in essence, the arithmetic decoder functions as a universal actuator, whose behavior can be completely configured by a choice of action model.

Reshaping of the action space. As alluded to before, the effect of the action model is to reshape the action space, which the following example will make clear. Figure 4.2 shows an illustrative example of the behavior of a binary arithmetic decoder equipped with an action model based on a Gated Linear Network-based context mixing language model (Veness, Lattimore et al., 2019) that has been pre-trained on 9MB of grandmaster games in PGN (Portable Game Notation) format, though this could be done with any (large) language model. On the left hand side of the table, we have the input to the decoder, and on the righthand side we have the decoded output; if we consider the first row, the LHS corresponds to the bitstring $10 = C_\rho(\text{a6})$ and the RHS corresponds to $D_\rho(10)$, with ρ here denoting our pre-trained language model. The LHS of the first 4 rows shows the encoding of a natural sequence of continuing moves (known as the Morphy Defense), while the last four rows show an illogical continuation of moves which ignore development, lose castling rights, and even hang the queen. One can see that much shorter codes are assigned to the more logical

sequence of moves. This shows the effect of the action model as providing a type of inductive bias, which we will discuss in greater depth later.

In contrast, one could also consider the effect of a completely uninformative action model, $\rho_{\text{UNIFORM}}(a|s) := 1 \,/\, |\mathcal{A}|$, which assigns uniform probability mass to each possible external action in every state. Here every single action would have the same codelength of $\lceil \log |\mathcal{A}| \rceil$, which would correspond to a naive binarization of the external action space.

Input bits	**Decoded Output**
10	a6
10010	a6 Ba4
100100	a6 Ba4 Nf6
1001010111	a6 Ba4 Nf6 O-O
010010101010011	Nh6
01001010101010000110000010	Nh6 Kf1
010010101010100011000001100100101	Nh6 Kf1 Qg5
0100101010101000110000011001001010101010010010001	Nh6 Kf1 Qg5 Na3

Figure 4.2: Arithmetic decoding example. Some examples of decoded outputs from a pre-trained model on chess, with the model's context set to the Ruy Lopez opening, namely: *e4 e5 Nf3 Nc6 Bb5*.

When to stop decoding. Figure 4.2 also highlights a technical issue which we need to resolve, namely, how and when is a decoded action to be transmitted to the external environment? For example, if we wanted a chess-playing agent whose action space was the space of single moves, we need some way to know when our decoded output should be communicated to the environment as an external action. Although other solutions are possible, in this work we adopt the convention that every external action can be described as a string formed by the concatenation of atomic symbols from a common alphabet. More formally, we assume that the action space $\mathcal{A} \subseteq \mathbb{A}^{\leq k}$, where $\mathbb{A}$ denotes the sub-action alphabet, and k is a positive constant. We assume that the sub-action alphabet always contains a privileged termination symbol $\top \in \mathbb{A}$, which has the semantics that when it is decoded it causes an external action to be communicated to the environment. Note that in finite action/state MDPs, this modification does not impose any restrictions nor add further expressive power. Returning to the example shown in Figure 4.2, by identifying the space character with $\top$, we would know when to transmit an external action. This is implemented formally via a function $\tau : \mathbb{A}^{\leq k} \to \mathcal{A}$ which takes actions and returns the action component up to but

not including the first $\top$, for example $\tau(\text{a6}\,\top) = \text{a6}$. This importantly handles the case of multiple $\top$ symbols, for example $\tau(\text{a6}\,\top\,\text{Ba4}\,\top) = \text{a6}$.

A terminal symbol is not the only way to know when to stop decoding. Another approach could be to only allow prefix-free codes. This will however run into it's own problems, such as what prefix-free encoding to use, how to enumerate the elements of $\mathcal{A}$ so that the corresponding prefix code can be found easily (and vice versa). Using an "optimal" prefix code would require the use of universal Turing machines and is beyond the scope of this chapter. Another choice to stop decoding is to consider the action before the last $\top$ symbol, instead of before the first. In this case the agent may take multiple actions without knowing the state in between them.

Internal action loop. External action selection is determined by executing our internal policy π until the concatenation of these binary actions uniquely decodes into an external action. Once the action model and arithmetic decoder have generated an external action, this external action will be sent to the external environment. The external environment will then return an external observation/reward to the action model and arithmetic decoder combination, and the internal policy receives a reward r_t from the external environment. This interaction is displayed graphically in Figure 4.1 and described procedurally by Algorithm 1.

Algorithm 1 Internal Agent-Environment loop

Require: Internal policy $\pi : \mathcal{S} \times \mathbb{B}^* \to \Delta\mathbb{B}$
Require: External environment $\mu : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{R})$
Require: Action model $\rho : \mathcal{S} \to \Delta\mathbb{A}^{\leq k}$

for $t = 1, 2, 3, \ldots$ **do**
  Observe $s_t, r_t \sim \mu(\cdot, \cdot | s_{t-1}, a_{t-1})$
  $a_t, q = \epsilon$
  while $\top \notin a_t$ **do**
    $b \sim \pi(\cdot | s_t, q)$
    $q = qb$
    $a_t = D_\rho(q | s_t)$
    $r_t = 0$
  $a_t = \tau(a_t)$
  Act a_t

Example. We conclude this section with an example execution in the context of our previous chess example. Here the state space is the finite history of moves. At the first time-step, the system receives an observation from the external environment. This observation becomes the current state $s_0 = \text{e4 e5 Nf3 Nc6 Bb5}$. Then given this state the internal policy takes internal binary actions one by one $b_0 = 1, b_1 = 0$. The internal policy continues to take internal binary actions until the sequence of internal binary actions uniquely decodes into an external action a_t. A concatenation of characters is an external action when it has a "terminal"

symbol $\top$, a6 $= D_\rho(b_0 b_1 | s_0)$. Now that we know the external action, it is sent to the external environment μ, $a_1 =$ a6 . Then we receive the next observation from μ,

$$o_1 \sim \mu(\cdot | s_0 a_1) = \mu(\cdot | \text{e4 e5 Nf3 Nc6 Bb5 a6})$$

and now $s_1 := s_0 a_1 o_1$ and the process repeats.

4.4 Connecting External to Internal Agents and Environments

In this section, we will describe formally how to augment an arbitrary external environment to an internal environment that the internal action agent is able to interact with; additionally if the external environment is Markovian then the internal environment will also be Markovian. Our approach will be to construct an augmented environment ϑ, called the internal environment, comprised of the true environment, the action model and the arithmetic decoder. We will also show how the internal action agent can be uplifted to an external agent, and then show that both the internal environment with an internal action agent and the external environment with the uplifted policy are equivalent in the sense that they achieve identical action-value functions. These results will allow for easier analysis of the internal action agent setup, as well as the ability to apply any result or algorithm specific to MDPs to the internal agent setup.

Internal Environment. The internal environment ϑ is a stochastic function over internal states and internal actions to internal states and rewards. The internal state space used here will be $\mathcal{I} := \mathcal{S} \times \mathbb{B}^{\leq n}$, the state from the external environment and previous internal actions taken by the internal agent, until they are decoded to an external action. We consider the finite set $\mathbb{B}^{\leq n}$ over the infinite set $\mathbb{B}^*$, as for any external action a with $\rho(a) > 0$ there will always be a finite number of binary actions needed to decode a; n is the maximum of those finite numbers. We will use $\top$ to denote the "terminal" symbol, that is, the symbol that indicates when the concatenation of internal actions corresponds to a complete external action, and is sent to the external environment. We will use the symbols s, s' for elements of $\mathcal{S}$, the first component of the internal state. We will use q, q' for elements of $\mathbb{B}^{\leq n}$, the second component of the internal state, the internal agent's previous internal actions. The symbol b will be used for the internal agent's internal action. The symbol a will be used for a decoded external action, e.g. $D_\rho(qb|s) = a$. The true external environment will be denoted by μ, which is a stochastic function from external states and external actions to external states and rewards. The external state space is $\mathcal{S}$. The external action space is $\mathcal{A} \subseteq \mathbb{A}^{\leq k}$.

Definition 4.1 (MDP (Internal) Environment ϑ). *The internal policy π interacts with an internal environment $\vartheta : \mathcal{I} \times \mathbb{B} \to \Delta(\mathcal{I} \times \mathcal{R})$ which is defined by the action model ρ (encoder/decoder C_ρ/D_ρ*

generated by ρ) and the true external environment μ as follows:

$$\vartheta(s'q'r|sq,b) := \begin{cases} \mu(s'r|s,\tau(a)) & \text{if } q' = \epsilon \ \wedge (\top \in a), \\ 1 & \text{if } s' = s \ \wedge q' = qb \wedge r = 0 \ \wedge (\top \notin a), \\ 0 & \text{otherwise} \end{cases}$$

where $a := D_\rho(qb|s)$, $(s'q', r) \in \mathcal{I} \times \mathcal{R}$, $sq \in \mathcal{I}$ and $b \in \mathbb{B}$.

The definition of ϑ is split up into three cases: In the first case the decoded qb contains the symbol $\top$, $\top \in a$ where $a := D_\rho(qb|s)$, and the previous binary characters q' resets to being the empty string ϵ. In this case the τ of the decoded action $D_\rho(qb|s)$ is sent to the external environment μ, and the next state s', is the external state s'. The second case of ϑ is when the internal agent is still decoding, that is, $\top \notin a$ and the next state $s'q' = sqb$ is updated by the agent's action b, and the internal reward r is 0. In the third case, where neither set of above conditions is satisfied, the probability of the state $s'q'$ and reward r is 0. In this way the environment ϑ is deterministic during the decoding process, and only stochastic when it sends the decoded action to the external environment.

Given the internal agent's policy π and the arithmetic decoder D_ρ, we can construct an external policy Π which will interact with the true external environment μ. The external policy Π is a stochastic function from external states $s \in \mathcal{S}$ to external actions $a \in \mathcal{A}$. To construct Π, we consider all possible binary strings $q \in \mathbb{B}^{\leq n}$ such that the arithmetic decoder will decode q into a given s. For this we will need to define a *decodable* subset of $\mathbb{B}^{\leq n}$. We will use $\mathbb{D}$ to denote the set of decodable binary strings. A string q is decodable if $\top$ is in the decoding of the string, and $\top$ is not in the decoding of the first $|q| - 1$ elements of the string. Formally this means

$$\mathbb{D}_s := \left\{ q \in \mathbb{B}^{\leq n} : \top \in D_\rho(q|s) \ \wedge \ \top \notin D_\rho(q_{<|q|}|s) \right\}.$$

We then consider the probability that π will output the internal binary actions that eventually construct q, which using the chain rule we can write as the product of probabilities that π will take the action of each element of q given the previous elements of q. All together this is written as follows:

$$\Pi(a|s) := \sum_{\substack{q \in \mathbb{D}_s: \\ a = \tau(D_\rho(q|s))}} \prod_{i=1}^{|q|} \pi(q_i | s q_{<i}). \tag{4.1}$$

It is important to note that there may be more than one binary string $q \in \mathbb{D}_s$ such that $a = \tau(D_\rho(q|s))$; this comes from how arithmetic decoders work. For example, consider a case where

$$\begin{aligned} &D_\rho(10|s) = e, \ D_\rho(100|s) = e4, \ D_\rho(101|s) = e4 \\ &D_\rho(1000|s) = e4\ c5, \ D_\rho(1001|s) = e4\ \top \\ &D_\rho(1010|s) = e4\ \top, \ D_\rho(1011|s) = e4\ e5 \end{aligned}$$

We have that both 1001 and 1010 are elements of $\mathbb{D}_s$ and both $\tau(D_\rho(1001|s)) = e4$ and $\tau(D_\rho(1010|s)) = e4$, therefore $\Pi(e4|s)$ would be a sum over 1001 and 1010.

Self-consistency of internal and external Q-values. We can use the external agent Π to interact with the external environment μ, just as any regular RL agent would.

Theorem 4.2 (Internal/External value equivalence)**.** *For all states $s \in \mathcal{S}$, previous internal actions $q \in \mathbb{B}^{\leq n}$, external actions $a \in \mathcal{A}$ and internal actions $b \in \mathbb{B}$, if $\tau(D_\rho(qb|s)) = a$ then*

$$Q^{\Pi}_{\mu}(s,a) = Q^{\pi}_{\vartheta}(sq,b). \tag{4.2}$$

That is, the action-value function for the external policy Π and external environment μ is equal to the action-value function for the internal policy π and the internal environment ϑ.

Proof. Instead of expanding the entire action-value function we will show that a single interaction between μ and Π are equal to an interaction between ϑ and π. Let s', r, a' be arbitrary states, rewards and actions, then

$$\begin{aligned}
\mu(s'r|s,a)\Pi(a'|s') &\overset{(a)}{=} \mu(s'r|s,a) \sum_{\substack{q' \in \mathbb{D}_{s'}: \\ a' = \tau(D_\rho(q'|s'))}} \prod_{i=1}^{|q'|} \pi(q'_i|s'q'_{<i}) \\
&\overset{(b)}{=} \vartheta(s'\epsilon r|sq,b) \sum_{\substack{q' \in \mathbb{D}_{s'}: \\ a' = \tau(D_\rho(q'|s'))}} \prod_{i=1}^{|q'|} \pi(q'_i|s'q'_{<i}) \\
&\overset{(c)}{=} \sum_{\substack{q' \in \mathbb{D}_{s'}: \\ a' = \tau(D_\rho(q'|s'))}} \vartheta(s'\epsilon r|sq,b)\pi(q'_1|s'\epsilon) \times \\
&\qquad \prod_{i=2}^{|q'|} \vartheta(s'q'_{<i}0|s'q'_{<i-1}, q'_{i-1})\pi(q'_i|s'q'_{<i}).
\end{aligned}$$

Step (a) comes from Equation 4.1; step (b) comes from the first case of Definition 5.1; step (c) comes from the second case of Definition 5.1, where $\vartheta(s'q'_{<i}0|s'q'_{<i-1}, q'_{i-1}) = 1$.

Therefore in the expansion of the action-value function we can replace one with the other,

$$
\begin{aligned}
&Q^{\Pi}_{\mu}(s,a)\\
&= \mathbb{E}^{\Pi}_{\mu}\left[\sum_{t=1}^{m} r_t | s,a\right]\\
&= \sum_{s_1,r_1,a_1} \mu(s_1 r_1|s,a)\Pi(a_1|s_1)\ldots \sum_{s_m,r_m,a_m} \mu(s_m r_m|s_{m-1},a_{m-1})\Pi(a_m|s_m)\sum_{t=1}^{m} r_t\\
&= \sum_{s_1,r_1,a_1}\left(\sum_{\substack{q'\in\mathbb{D}_{s_1}:\\ a_1=\tau(D_\rho(q'|s_1))}} \vartheta(s_1\epsilon r_1|sq,b)\pi(q'_1|s_1\epsilon)\prod_{i=2}^{|q'|}\vartheta(s_1 q'_{<i}0|s_1 q'_{<i-1},q'_{i-1})\pi(q'_i|s_1 q'_{<i})\right)\\
&\ldots \sum_{s_m,r_m,a_m}\\
&\left(\sum_{\substack{q'\in\mathbb{D}_{s_m}:\\ a_m=\tau(D_\rho(q'|s_m))}} \vartheta(s_m\epsilon r_1|s_{m-1}\epsilon,b)\pi(q'_1|s_m\epsilon)\prod_{i=2}^{|q'|}\vartheta(s_m q'_{<i}0|s_m q'_{<i-1},q'_{i-1})\pi(q'_i|s_m q'_{<i})\right)\sum_{t=1}^{m} r_t\\
&= \mathbb{E}^{\pi}_{\vartheta}\left[\sum_{t=1}^{m} r_t|sq,b\right] = Q^{\pi}_{\vartheta}(sq,b)
\end{aligned}
$$

Notice that the horizon is defined externally, i.e. an m-horizon return in the external environment corresponds to m decoded external actions, and not the number of internal actions taken. □

Because of Equation 5.4 we are able to say that if an internal agent π performs well, in the sense of a high action-value, in the internal environment ϑ, then the uplifted version of the agent Π, performs well in the true external environment μ.

4.5 A Universal Action Interface for Multi-task RL

A key complication and limiting factor in the design of any multi-task RL system is how to deal with the potentially radically different action spaces required for each distinct task. While it is feasible to make a generic agent work well across multiple similar domains e.g. Atari games (Mnih, Kavukcuoglu et al., 2015), the situation becomes considerably more complicated when the action spaces of the different tasks vary dramatically. The arithmetic encoding-based approach we advocate provides an elegant solution to this problem, which builds on techniques from universal source coding.

Given $K > 1$ coding distributions, it is straightforward to combine them into a universal ensemble whose compression performance will be close to that of the best coding distribution in hindsight. If we denote the ith coding distribution by ρ_i, one can take a uniform Bayesian mixture of the K coding distributions, whose marginal distribution over sequences is given

by

$$\xi(x_{1:n}) := \sum_{i=1}^{K} \frac{1}{K} \rho_i(x_{1:n}). \tag{4.3}$$

A standard dominance argument shows that the logarithmic loss/coding length of the mixture ξ compared to any choice of action model j is bounded by

$$\begin{aligned} -\log \xi(x_{1:n}) &\leq -\log\left(\frac{1}{K}\rho_j(x_{1:n})\right) \\ &= -\log \rho_j(x_{1:n}) + \log K, \end{aligned}$$

or in other words, the excess log-loss is bounded by a constant, which is asymptotically negligible when one considers the time-averaged performance of the ensemble.

This has important ramifications for multi-task reinforcement learning in our internal action formulation. Recently, various works (Janner et al., 2021) have attempted to frame reinforcement learning in terms of probabilistic sequence models over interaction strings, i.e. defining a sequential probability measure ν over strings that represent state/reward/action histories in the form $s_1 r_1 a_1 \ldots$. By taking a uniform Bayesian mixture over multiple instances of these history-based measures for different tasks, just as in the coding distribution example, one also obtains a sequence model that is universal across all of these tasks. More formally, given a history string h which is an element of $(\mathcal{S} \times \mathcal{R} \times \mathcal{A})^* \cup ((\mathcal{S} \times \mathcal{R} \times \mathcal{A})^* \times (\mathcal{S} \times \mathcal{R}))$, we can define the uniform Bayesian mixture

$$\xi(h) = \sum_{i=1}^{K} \frac{1}{K} \nu_i(h) \tag{4.4}$$

over K history based measures ν_i, with each ν_i corresponding to a task specific history model. Note that this formulation in terms of measures on strings still implies the usual Bayesian learning in terms of sequential updating of the posterior, it is just hidden in this notation; see Section 2 by Milan et al. (2016) for a brief overview.

An interesting effect now emerges if we use the conditional action distribution $\xi(\cdot|s_1 r_1 a_1, \ldots, s_n r_n)$ as the action model in our setup. In particular, this action model will rapidly learn to *automatically generate actions appropriate for the underlying task*, without requiring any task identity information. How this works is subtle; Bayesian inference is used implicitly by ξ to determine which task the agent is most likely in, and due to the rapid convergence of the Bayesian mixture to the best task specific model, the action model used for decoding after a small number of external environment interactions will essentially behave the same as if we knew which task specific action model to use in the first place. In other words, what this means in practice is that one can use ξ as the action model, and C_ξ will produce codes which are almost as short as any task-specific action encoding C_{ρ_j}. In particular, this implies that short bitstrings can decode to very different external actions which are plausible under either task-specific model.

The most interesting aspect about this construction is that the internal action formalism

allows us to treat a multi-task reinforcement problem as a single reinforcement learning task with a common action space. Compared to the union over several different action spaces, in our setup the agent essentially has a weight on the more likely actions meaning it doesn't need to consider all the possible actions at each time, which would be the case if the action space was the union of the different action spaces.

4.6 Discussion

This section discusses some interesting and potentially surprising ramifications of information-theoretic internal actions.

Uninformative internal policy π and application to Large Language Models. An interesting corollary of the internal reinforcement learning setup is that a uniform policy over internal actions gives rise to an external agent that selects actions that are essentially distributed to the action model. This is a by-product of the duality between optimal codes and probabilities, and can be seen with the following argument. In the case of a binary internal action space, a uniform policy will generate a particular sequence $b_{1:n}$ with probability 2^{-n}. Now, notice that the probability of an action $a \in \mathcal{A}$ in state s where $C_\rho(a|s) = b_{1:n}$ is given by

$$\rho(a|s) = 2^{\log_2 \rho(a|s)} \approx 2^{-|C_\rho(a|s)|} = 2^{-n}.$$

The approximate equality step is due to the small gap between the optimal code length $-\log_2 \rho(a|s)$ and the realised code length $C_\rho(a|s)$ produced by an arithmetic encoder coupled to ρ. The size of this gap is bounded by 1, but is essentially negligible for the purposes of this argument.

This has interesting ramifications for constructing agents when the action model is already useful, such as in the case of Large Language Models (LLMs) obtained by supervised learning on massive amounts of internet data. Random behaviour by the internal policy in this case will still produce useful behavior, which provides a natural starting point for any internal policy learning technique. In particular random internal-action exploration becomes targeted, possibly leading to optimal external-action exploration, in a similar fashion to Thompson Sampling. It is in this way that the action model provides a powerful mechanism for specifying data-dependent prior knowledge to existing reinforcement learning algorithms.

Specifying the action space from data. In complicated environments, it may be difficult or complicated to precisely specify the action space explicitly. This situation readily arises in natural language domains for example. In these cases it is more natural to simply learn a probabilistic model of the domain. Our internal agent formalism directly allows for this possibility via the action model. The action model allows for a strict separation between pre-training on data, for example pre-training an action model using a collection of grandmaster games in chess, and the resultant learning behavior of the internal agent.

It is also worth pointing out an interesting connection to meta-learning with sequence models across many tasks. Perhaps surprisingly, perplexity-based meta-learning of history-dependent LLMs is closely related to the explicit Bayesian mixture solution described in Equation 4.4. In particular, one can show that in many standard meta-learning setups, the optimal perplexity-minimizing solution is *exactly* a Bayesian mixture distribution (Ortega, Wang et al., 2019). Provided that a sufficiently powerful history-dependent model is used (such as the case with LLMs based on Transformers) to model the interaction histories, a low-perplexity solution can be seen as a learnt approximation to the explicit Bayesian construction we provided in in Equation 4.4. In this way the action space for a multi-task agent can be learnt directly from data alone, which goes some way to explaining the recent empirical success of approaches such as Janner et al. (2021). In other words, if one wanted an agent that could play both chess and something with a radically different action space such as the text-based NetHack (Küttler et al., 2020), a natural strategy would be to pre-train using meta-learning a sequence model from example trajectories in both games, and use this to define the action model; the internal action framework will then automatically deal with the different underlying action spaces.

Comparison to binarization. It is instructive to consider the differences between a direct binarization of the action space compared with our approach. One can interpret the combination of an action model and a binary arithmetic decoder as a generalized form of binarization. As discussed earlier, an action model which assigned a uniform distribution over the action space in every state is equivalent to a naive binarization, with every action being assigned a code-length of $\lceil \log_2 |\mathcal{A}| \rceil$. Binarization of actions in reinforcement learning has the obvious benefit of reducing the size of the action space, which can lead to some benefits (Majeed and Hutter, 2021). However, often this binarization comes with a corresponding increase to the planning horizon, and in many circumstances provides no benefits.

Our non-uniform binarization essentially reshapes the action space according to the knowledge contained within the action model. Thus planning using various types of depth limited search takes on a different meaning in our internal reinforcement learning setting. Although the planning algorithm may only be searching d steps ahead, the implied information-theoretic planning horizon might be much greater than d.

Computational advantages. Many reinforcement learning techniques require an ability to efficiently generate a random sample from the action space. A convenient property of our formalism is that it provides a generic technique to generate samples from arbitrary action models/action spaces. This is a byproduct of having an arithmetic decoder coupled to an action model. By generating a sequence of bits $y_{1:m}$ with each bit sampled from a Bernoulli$(1/2)$ distribution, and feeding them to a binary arithmetic decoder D_ρ coupled to the action model ρ, one can show that external action $a := D_\rho(y_{1:m})$ is distributed according to ρ MacKay (2003), which resembles Thompson sampling. We can also efficiently compute the probability of a as a product of conditional probabilities ($P[a] = \prod_{t=1}^{k} P[b_t|b_{<t}]$) required for some learning algorithms. We can even efficiently compute the cumulative probability

based on the recursion $P(X_{1:k} \leq b_{1:k}) = [\![b_1 = 1]\!]P(X_1 = 0) + P(X_{2:k} \leq b_{2:k}|X_1 = b_1)P(X_1 = b_1)$. Unfortunately binarization does *not* lend itself to an efficient way of computing the most probable (MAP) action $\arg\max_a P(a)$. But Thompson sampling for large spaces often performs better than MAP anyway, since the latter is not representation invariant and favors brittle solutions. Binarization decreases the branching factor in planning algorithms but increases the planning horizon. Since binarized actions are length-optimized this *may* still lead to a net win. For example, depth-limited planning techniques, which typically have an exponential dependence on the length of the horizon, now have an exponential dependence on the combined code-length under ρ, which drastically alters their semantics and is closely connected to using a prior policy to guide search such as in successful approaches for Computer Go (Orseau and Lelis, 2021; Orseau, Lelis et al., 2018; Silver, Hubert et al., 2018).

Pre-training and universality. A common use case in machine learning is to consider fine tuning an existing pre-trained model to save on compute. The next result shows that pre-training on any data will not affect the asymptotic performance of any consistent density estimator. In our context, it suggests that a good general approach to constructing an action model for a new domain might be to first pre-train on large, task-agnostic data and then to use fine tuning to incorporate task-specific knowledge if this data is available.

More formally, consider sequences $X_{1:\infty}$ over a finite alphabet $\mathcal{X}$ sampled from P_{θ_0}. Assume $\theta(X_{1:n})$ is a consistent estimator of θ_0. Then whatever the first k samples $x_{1:k}$ are, $\theta(x_{1:k}X_{k+1:n})$ is still a consistent estimator of θ_0. Additionally, the reverse is also true. Most importantly, this holds without *any* assumptions on the stochastic process $(X_t) \sim P_{\theta_0}$.

Proposition 4.3 (consistency is immortal). *For any fixed $k \in \mathbb{N}$, $\theta(X_{1:n})$ is a consistent estimator of θ_0 if and only if for all $x_{1:k}$ such that $P_{\theta_0}[x_{1:k}] > 0$, $\theta(x_{1:k}X_{k+1:n})$ is a consistent estimator of θ_0.*

Proof. ($\Rightarrow$)

Let $\mathcal{N} := \{x_{1:\infty} : \theta(x_{1:n}) \not\to \theta_0\}$ be the set of sequences on which convergence fails. The consistency assumption implies

$$\begin{aligned} 0 &= P_{\theta_0}[\mathcal{N}] \\ &= \sum_{x_{1:k} \in \mathcal{X}^k} P_{\theta_0}[\mathcal{N}|x_{1:k}]P_{\theta_0}[x_{1:k}] \\ &\geq P_{\theta_0}[\mathcal{N}|x_{1:k}]P_{\theta_0}[x_{1:k}]. \end{aligned}$$

Since $P_{\theta_0}[x_{1:k}] > 0$ by assumption, the RHS can only be 0 if $P_{\theta_0}[\mathcal{N}|x_{1:k}] = 0$, which implies $P_{\theta_0}[x_{1:k} \times \mathcal{X}^\infty \setminus \mathcal{N}|x_{1:k}] = 1$, hence $\theta(x_{1:k}X_{k+1:n}) \to \theta_0$ with $P_{\theta_0}[\cdot|x_{1:k}]$-probability 1.

($\Leftarrow$)

Again let $\mathcal{N} := \{x_{1:\infty} : \theta(X_{1:n}) \not\to \theta_0\}$ be the set of sequences on which convergence fails. The consistency assumption implies that for all $x_{1:k}$ for which $P_{\theta_0}[x_{1:k}] > 0$, we have

$P_{\theta_0}[x_{1:k} \times \mathcal{X}^\infty \setminus \mathcal{N} | x_{1:k}] = 1$, which implies that $P_{\theta_0}[\mathcal{N}|x_{1:k}] = 0$. Therefore,

$$P_{\theta_0}[\mathcal{N}] = \sum_{x_{1:k} \in \mathcal{X}^k : P_{\theta_0}[x_{1:k}] > 0} P_{\theta_0}[\mathcal{N}|x_{1:k}] P_{\theta_0}[x_{1:k}] = 0$$

which implies $P_{\theta_0}[\mathcal{X}^\infty \setminus \mathcal{N}] = 1$, hence $\theta(X_{1:n}) \to \theta_0$ with P_{θ_0}-probability 1. □

One consequence of this proposition is that for a given ρ and π, if Π defined in Equation 4.1 is a consistent estimator of π^*_μ, the optimal policy in μ, then if the action model ρ is pre-trained on additional data then Π is still a consistent estimator.

Discounting. One subtlety that arises is how to best communicate a reward from the environment to the internal policy. Notice that in Algorithm 1, after the first internal action, r_t is set to 0. This has the effect of preserving the return and associated discounting schedule in the external environment. Although in the case of uninformative action binarization one can map a discounted external setup to an equivalent discounted internal setup (Majeed and Hutter, 2021), attempting a more general construction along those lines in our case requires time-dependent, and worse, history-dependent discounting which runs into both technical and computational challenges.

4.7 Related Work

We now discuss some related work.

Compression-based RL. Using compression to aid with machine learning goes back to at least the work by Frank et al. (2000), where compression-based models were compared to classical machine learning methods on a number of natural language problems. More recently, Hamilton et al. (2013) used compression with Predictive State Representation on domains with large observation spaces to aid with the intractability. Botvinick et al. (2015) discussed the internal representation of reinforcement learning, specifically a natural or efficient coding of the internal representation. Veness, Bellemare et al. (2015) used compression-based techniques for policy evaluation via action-value estimation.

Large transformer/language models used to aid RL. Language models have had a recent resurgence, starting with *Attention is all you need* (Vaswani et al., 2017) and being followed by the success of GPT-2 (Radford et al., 2019) and GPT-3 (Brown, Mann et al., 2020). Despite the accomplishment of language models on NLP tasks, there have only been a few circumstances where these techniques have translated to the field of reinforcement learning. Luketina et al. (2019) provides a survey of reinforcement learning methods which have been improved with the addition of natural language approaches. One such example is Kaplan et al. (2017), where natural language methods were used with deep reinforcement learning to play Atari games. In addition to this, recent work on using transformer models with reinforcement learning includes: Parisotto et al. (2020), Noever et al. (2020) train GPT-2 (Radford et al., 2019) on the

PGN format to learn chess, Ciolino et al. (2020) trained GPT-2 in a similar way to learn Go, and Stein et al. (2020) used Transformers for Deep Q-learning to play Atari games. Krause et al. (2020) introduced a coding scheme to improve small language models.

4.8 Future Work

Arithmetic encoding has a number of extensions which deserve further investigation in the context of reinforcement learning. In particular, one can generalise arithmetic encoding to time-adaptive coding distributions, which is known as adaptive arithmetic encoding. While one could crudely incorporate this notion into our existing work, a more complete treatment would require going outside the MDP formalism.

Finally from a theory perspective, there are some additional generalizations that can be made, though they are beyond the scope of this chapter. These include a more thorough treatment of the infinite horizon case by using discounting, investigating setups which do not require an end of action symbol $\top$ (as discussed earlier) and allowing the external action space to be countably infinite.

4.9 Conclusion

In this work we have laid the conceptual foundations for information-theoretic actuation. We revisited the meaning of action in reinforcement learning, and explored a particular type of internal viewpoint. We have demonstrated how our method is theoretically well justified, by formally connecting it to a traditional reinforcement learning MDP setup. We argued that such a framework is well positioned to take advantage of the recent progress in large sequence/language models for multitask reinforcement learning problems over large action spaces. The next step is to explore application of this formalism in conjunction with modern sequence modelling techniques on some benchmark problems to better understand the potentials and limitations of this approach. Multi-task RL problems with vastly different action spaces seem the most natural setting where our approach could have immediate impact.

Chapter 5

Compress and Converse

This chapter is in preparation for publication as

E. Catt, J. Veness et al. (2022a). *Compress and Converse: An information-theoretic approach to reinforcement learning.*

Abstract

Large language models, particularly those based on transformer models, have had much success in the field of natural language processing. A natural question is then, how can we utilise these advances for reinforcement learning? In this work we argue for an information-theoretic approach, leveraging previous work that, with the help of an action model, is able to binarise the action space and an information-theoretic policy evaluation algorithm which can take advantage of this binarisation. We show that this approach is theoretically well-founded.

5.1 Introduction

While the generation of plausible conditional continuations of text has many immediate potential applications, a language model is in some sense just a language model; without a sense of goal, it is merely a very sophisticated tool whose limited sense of agency is a byproduct of how it is applied in any given use case. Furthermore, the current best models are statically trained and then deployed, which fundamentally limits their ability to adapt to new input in a timely fashion, ultimately limiting their utility. There is also a crisis in evaluating these methods, given the unequal distribution of computational resources, the use of private datasets for training, and the prevalence of performance metrics which do not properly take into account the size of the model when comparing performance. Despite this, simply by looking at the generated text and performing a qualitative evaluation, it is clear that this next generation of models goes well beyond what has ever been achieved before.

A natural question then is how to best adapt these large, pre-trained models to specific goal-directed tasks in a flexible and general manner. This work attempts to take a step towards making these models more generally useful, by a unification of reinforcement learning with information theory (Hutter, 2005). In particular, we propose a mechanical transformation of an adaptive language model into a reinforcement learning agent. The key technique it builds on is the Compress and Control algorithm (CnC) (Veness, Bellemare et al., 2015), which provides a means for converting any compression algorithm into a policy evaluation algorithm for MDP-based reinforcement learning. A key caveat with this technique is the linear time and space dependence on the number of possible actions. While this method was shown to be sufficient for converting Lempel-Ziv (Ziv and Lempel, 1977) (with Zip being a popular use case) into an agent which can play simple Atari games well, it would be unrealistic to think the method could deal with an action space consisting of the space of all possible words, let alone sentences and paragraphs. The key technical challenge we overcome in this chapter is to use a novel application of arithmetic decoding to both reduce the size of the action space needed for the Compress and Control technique, as well as provide a means to leverage the latest advances in large, static, pre-trained language models.

Content. The chapter is structured as follows: Section 5.2 we go over the required concepts and notation of this chapter; Section 5.3 we review Compress and Control and information-theoretic actuation. Section 5.4 we show how the two above concepts can be utilised together; Section 5.5 we examine modification and consequences of our approach; Section 5.6 we go over possible extensions; Section 5.7 we conclude and give our closing thoughts.

5.2 Preliminaries

In this chapter, we follow the notation of Chapter 4 with the following additions:

Markov Chains. A Homogenous Markov Chain (HMC) given by the stochastic process $(X_t)_{t\geq 0}$ over state space $\mathcal{X}$ is said to be:

- (AP) aperiodic iff $gcd\{n \geq 1 : \mathbb{P}(X_n = x | X_0 = x) > 0\} = 1, \forall x \in \mathcal{X}$
- (PR) positive recurrent iff $\mathbb{E}[\min\{n \geq 1 : X_n = x\} | X_0 = x] < \infty, \forall x \in \mathcal{X}$
- (IR) irreducible iff $\forall x, x' \; \exists n \geq 1 : \mathbb{P}(X_n = x' | X_0 = x) > 0$
- (EA) essentially aperiodic iff $\forall x \in \mathcal{X}$ either $gcd\{n \geq 1 : \mathbb{P}(X_n = x | X_0 = x) > 0\} = 1$ or $\{n \geq 1 : \mathbb{P}(X_n = x | X_0 = x) > 0\} = \emptyset$

Note also that EA+IR implies AP.

5.3 Background

In this section, we will give an overview of two methods which we unite in this chapter: Compress and Control and Information-Theoretic Actuation.

5.3.1 Generative Policy Evaluation in MDPs

In MDP-based reinforcement learning, policy evaluation typically refers to the task of estimating the value function implied by a given choice of stationary policy. One popular approach is to combine various forms of function approximation with techniques such as Temporal Difference learning (Sutton, 1988) or Monte Carlo estimation to learn value function estimates directly from data generated by direct execution of the policy. There is however a relatively under-explored different family of methods which exploit a form of duality in value function representation (Wang et al., 2007); the key idea is that the value function for a given policy can be obtained by modelling various stationary distributions that arise from a choice of stationary policy and environment. The major advantage of this perspective is that it enables the use of well-developed supervised learning techniques for modelling probability distributions to be applied to the task of policy evaluation.

To see how this could work in practice, recall that the combination of an MDP $\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mu)$ and stationary policy π will give rise to a time-homogeneous Markov chain over $\mathcal{S}$. Under appropriate conditions, this will give rise to a unique stationary distribution $d^{\pi}_{\mathcal{M}} : \mathcal{S} \to [0, 1]$, which is typically referred to as the *state-occupancy measure* in the literature. Once the chain has been executed for sufficiently long such that all transient effects have disappeared, the quantity $d^{\pi}_{\mathcal{M}}(s)$ gives the (limit) probability that the system formed by $\mathcal{M} + \pi$ will be in state s at any given sufficiently large time. One might then wonder whether stationary measures over richer objects than merely $\mathcal{S}$ are implied by such a quantity, and can they contain sufficient information such that the value function can be recovered from them in a tractable manner? The answer to both those questions is yes. The next section will give a concrete example of a particular dual value estimation algorithm, which we will later extend to the internal action reinforcement learning setting.

The main insight with dual reinforcement learning techniques is to learn a time-invariant stationary distribution which we can then use to obtain sufficient information to reconstruct the value function.

For the purposes of introduction we will give an overview here, with details of the exact construction following later in the document.

Decomposing the action-value function into stationary measures

Given a finite horizon m and the setup above, one can show that a unique stationary measure ν for the time-homogeneous Markov chain over the augmented state-space $(\mathcal{A}, \mathcal{S}, \mathcal{R})^{m+1}$ exists. In other words, given a realised interaction sequence of length $n \gg m$,

$$a_1, s_1, r_1, ..., a_{m+1}, s_{m+1}, r_{m+1}, \ldots, a_n, s_n, r_n,$$

obtained by executing a stationary policy π in $\mathcal{M}$, we can construct a set $\mathcal{I}$ of augmented states in the form

$$\begin{aligned}\mathcal{I} = \{&(a_1, s_1, r_1, ..., a_{m+1}, s_{m+1}, r_{m+1}),\\ &(a_2, s_2, r_2, ..., a_{m+2}, s_{m+2}, r_{m+2}), \ldots, (a_{n-m}, s_{n-m}, r_{n-m}, ..., a_n, s_n, r_n)\},\end{aligned}$$

which for sufficiently large n will be distributed according to ν. Assuming the return can be defined as a function of each augmented state, for example if $z_i := \sum_{j=0}^{m} r_{i+j}$, we can also consider it to also be part of the augmented state. Since ν exists, then its marginal distribution $\nu(a, s, z)$ exists, where a and s denote the first action and state in the augmented state space respectively, and z denotes the return. This marginal distribution contains all the information we need to define the value function, and can be conditioned on, decomposed and factored in various ways. For example, the probability of getting a return of z starting in state s and executing action a is simply $\nu(z \mid a, s)$, so

$$Q_\mu^\pi(s, a) = \sum_z z\, \nu(z \mid a, s) \tag{5.1}$$

follows immediately by definition. This form of learning a distribution over returns has first been introduced in Veness, Bellemare et al. (2015) before being popularized as Distributional RL in Bellemare, Dabney et al. (2017). By applying Bayes rule, one obtains the identity

$$\nu(z \mid a, s) = \frac{\nu(s \mid a, z)\, \nu(z \mid a)}{\sum_{z'} \nu(s \mid a, z')\, \nu(z' \mid a)}. \tag{5.2}$$

One reason to flip states with actions and returns is that state spaces are typically much larger than action spaces and under certain conditions return spaces, which makes learning $\nu(s|a, z)$ and $\nu(z|a)$ easier than $\nu(z|a, s)$. So to approximate the value function, one simply needs to model the terms $\hat{\nu}(s \mid a, z) \approx \nu(s \mid a, z)$ and $\hat{\nu}(z \mid a) \approx \nu(z \mid a)$, giving the following plug-in estimate of the action-value function

$$\hat{Q}_\mu^\pi(s, a) := \sum_{z \in \mathcal{Z}} z\, \frac{\hat{\nu}(s \mid a, z)\, \hat{\nu}(z \mid a)}{\sum_{z' \in \mathcal{Z}} \hat{\nu}(s \mid a, z') \hat{\nu}(z' \mid a)}. \tag{5.3}$$

Density estimation in practice

What remains now is to discuss some strategies for constructing our estimates $\hat{\nu}(s \mid a, z)$ and $\hat{\nu}(z \mid a)$ from data. By executing a stationary policy π in environment $\mathcal{M}$ for n steps, we can construct a dataset of action-state-return triples

$$\mathcal{D} = \{(a_1, s_1, z_1), (a_2, s_2, z_2), \ldots, (a_{n-m}, s_{n-m}, z_{n-m})\}$$

by simply recording each state-action pair and associated (delayed) return. Note that although only the finite-horizon case is considered here, extending it to the discounted reward setting can be done trivially using the standard approach of determining the effective horizon and truncating.

Bucketed Estimation with adaptive coding distributions

Given a dataset $\mathcal{D}$, we can define the following action and/or return filtered datasets by

$$\mathcal{D}_{a,z} = \{(a', s, z') \in \mathcal{D} \mid a' = a, z' = z\} \quad \text{and} \quad \mathcal{D}_a = \{(a', s, z) \in \mathcal{D} \mid a' = a\}.$$

Furthermore, from $\mathcal{D}_{a,z}$, we can define a string of states $s_{1:|\mathcal{D}_{a,z}|}$ formed by concatenating each state component of the action-state-return tuples in $\mathcal{D}_{a,z}$ in a fixed, but arbitrary order. Similarly, we can also define a string of returns $z_{1:|\mathcal{D}_a|}$ by concatenating the return components of $\mathcal{D}_a$. Using these strings, and a choice of a pair of adaptive coding distributions $\rho : \mathcal{S}^* \to [0, 1]$ and $\nu : \mathcal{Z}^* \to [0, 1]$, we can define the estimators

$$\hat{\nu}(s \mid a, z) := \rho\left(s \,||\, s_{1:|\mathcal{D}_{a,z}|}\right) \quad \text{and} \quad \hat{\nu}(z \mid a) := \rho\left(z \,||\, z_{1:|\mathcal{D}_a|}\right).$$

The $||$ indicates that $\rho(\cdot)$ has been learned from/trained on $s_{1:|\mathcal{D}_{a,z}|}$ respectivly $z_{1:|\mathcal{D}_a|}$. Bayesian learning is done by conditioning on past data, so $||$ is simple conditioning $\mid$ in this case. This was the approach taken by the CnC algorithm given in Veness, Bellemare et al. (2015), using a combination of techniques (Bellemare, Veness and Bowling, 2013; Bellemare, Veness and Talvitie, 2014; Hutter, 2013) for each coding distribution. Note that if the adaptive coding distribution permits efficient incremental updating, this naturally leads to an online action-value estimation scheme.

Compress and Control analysis

The decomposition of the action value in Equation 5.1 and Equation 5.2 may not always be well defined. In Veness, Bellemare et al. (2015) a set of necessary conditions were found. The conditions were that the environment μ and the policy π form a (IR+EA+PR) Homogenous Markov Chain. We will be using this result later in this work.

It has been shown that the approximation of $Q^\pi_\mu(s, a)$, described in Equation 5.3, converges to $Q^\pi_\mu(s, a)$, provided the estimators of ν are consistent estimators.

5.3.2 Information-Theoretic Actuation

More recently, an augmentation of the traditional reinforcement learning framework was proposed in Chapter 4 (Catt, Hutter et al., 2022). In this augmentation, the policy would always take binary actions that were filtered through an arithmetic decoder and action model. In this setup, the state space $\mathcal{S}$ was modified to a new state space $\widehat{\mathcal{S}} := \mathcal{S} \times \mathbb{B}^{<n}$ which comprised of the original state space and the previous binary action, and the modified (now binary) action space $\widehat{\mathcal{A}} = \mathbb{B}$. Under the new state and action spaces, together with an action model and arithmetic decoder, the internal environment was defined as follows:

Definition 5.1 (**MDP (Internal) Environment ϑ**). *The internal policy π interacts with an internal environment $\vartheta : \widehat{\mathcal{S}} \times \widehat{\mathcal{A}} \to \Delta(\widehat{\mathcal{S}} \times \mathcal{R})$ which is defined by the action model ρ (encoder/decoder C_ρ/D_ρ generated by ρ) and the true external environment μ as follows:*

$$\vartheta(s'q', r|sq, b) := \begin{cases} \mu(s', r|s, \tau(a)) & \text{if } q' = \epsilon \ \wedge (\top \in a), \\ 1 & \text{if } s' = s \ \wedge q' = qb \wedge r = 0 \ \wedge (\top \notin a), \\ 0 & \text{otherwise} \end{cases}$$

where $a := D_\rho(qb|s)$, $(s'q', r) \in \widehat{\mathcal{S}} \times \mathcal{R}$, $sq \in \widehat{\mathcal{S}}$ and $b \in \widehat{\mathcal{A}}$.

It was shown that this setup was equivalent to the traditional RL setup, at least in an action-value function sense.

Theorem 5.2 (Internal/External value equivalence). *For all states $s \in \mathcal{S}$, previous internal actions $q \in \widehat{\mathcal{A}}^{\leq n}$, external actions $a \in \mathcal{A}$ and internal actions $b \in \widehat{\mathcal{A}}$, if $\tau(D_\rho(qb|s)) = a$ then*

$$Q^\Pi_\mu(s, a) = Q^\pi_\vartheta(sq, b). \tag{5.4}$$

That is, the action-value function for the external policy Π and external environment μ is equal to the action-value function for the internal policy π and the internal environment ϑ.

The information-theoretic actuation augmentation was shown to have several advantages over the traditional RL setup. These include, but were not limited to: efficient sampling of the action space, the ability to implicitly learn multiple tasks and using a universal action model leading to a universal action space.

5.4 Compress and Converse

Compress and converse is the unification of Compress and Control policy evaluation with the binarised setup which we are provided with from information-theoretic actuation.

Similarly to compress and control, by showing the environment and policy lead to a (IR+EA+PR) HMC we can show that there exists a joint distribution ν over $\mathcal{Z} \times (\widehat{\mathcal{A}} \times \widehat{\mathcal{S}} \times \mathcal{R})^{m+1}$ such that

we can then express the action-value function as

$$Q^{\pi}_{\vartheta}(sq, b) = \sum_{z \in \mathcal{Z}} z\, \nu(z|sq, b) = \sum_{z \in \mathcal{Z}} z \frac{\nu(sq|z, b)\nu(z|b)}{\sum_{z' \in \mathcal{Z}} \nu(sq|z', b)\nu(z'|b)} \tag{5.5}$$

We can then define the (binary action) estimators $\hat{\nu}(z|b)$ and $\hat{\nu}(sq|z, b)$ of $\nu(z|b)$ and $\nu(sq|z, b)$ respectively. Then like Equation 5.3, we can use these estimators to build an approximation of the action-value function.

$$\hat{Q}^{\pi}_{\vartheta,n}(sq, b) := \sum_{z \in \mathcal{Z}} z\, w^{z,b}_{n}(sq) = \sum_{z \in \mathcal{Z}} z \frac{\hat{\nu}(sq \mid b, z)\, \hat{\nu}(z \mid b)}{\sum_{z' \in \mathcal{Z}} \hat{\nu}(sq \mid b, z')\hat{\nu}(z' \mid b)} \tag{5.6}$$

In the next section, we will show that the above decomposition is theoretically well-founded and that the approximation Equation 5.6 converges to the original (non-binary action) action-value.

5.4.1 Theoretical Results

In this section, we will demonstrate how using the internal action reinforcement learning setup with specific estimators can lead to an accurate approximation of the value function, which will eventually converge to the true value function. Thus we are justified in using the approximation of the value function for policy evaluation in reinforcement learning.

Our goal is to demonstrate that we can use the value function approximation of Equation 5.6 to accurately estimate the value function of the internal agent in the internal environment. We will do this by showing that the internal environment ϑ (Definition 5.1) together with a policy π forms a (IR+EA+PR) HMC. Once we have done this, we then decompose the value function and show that we are able to accurately approximate the original value function.

We need to show that ϑ with a stationary policy, a state space $\widehat{\mathcal{S}} = \mathcal{S} \times \mathbb{B}^{<n}$ and action space $\widehat{\mathcal{A}} = \mathbb{B}$, $\mathcal{M} = (\widehat{\mathcal{S}}, \widehat{\mathcal{A}}, \vartheta)$ forms a finite (IR+EA+PR) HMC. This may not be true in general, so we will have to make assumptions on the original environment μ and the action model ρ.

Definition 5.3. *Given ρ and s, a binary string p is decodable if it is not the empty string and $\top \in a$, where $a = D_{\rho}(p|s)$, and no prefix of p is decodable.*

We make a small assumption needed to prove the HMC result. This assumption is essentially just saying that there exist two external actions which have binary codes that are different in length by exactly one. This (or something similar) turns out to be a necessary if we want the HMC to be EA.

Assumption 5.4. *We assume that D_{ρ} is reasonable in the sense that there exist binary strings p, p' and $s \in \mathcal{S}$ such that p and p' are decodable and $|p| = |p'| + 1$.*

If we do not make Assumption 5.4, or some similar assumption, then we can show that the HMC from $\mathcal{M} = (\widehat{\mathcal{S}}, \widehat{\mathcal{A}}, \vartheta)$ will not be EA we show this with the example below.

Example 5.5. *Suppose that by the choice of ρ, the number of binary actions taken to decode an external action is always even. This means that the periodicity will always be 2, and therefore the resulting HMC will not be EA.*

Now we can state our HMC result for the binary action setting. If we assume the original process forms a HMC, then we can show that the binary action augmentation also forms a HMC.

Lemma 5.6. *Under Assumption 5.4, if $(\mathcal{S}, \mathcal{A}, \mu)$ forms a (IR+EA+PR) HMC, then the binary action $\mathcal{M} = (\widehat{\mathcal{S}}, \widehat{\mathcal{A}}, \vartheta)$ with stationary policy π forms a (IR+EA+PR) HMC.*

We will assume that the estimators $\hat{v}(z|b)$ and $\hat{v}(sq|z,b)$ are consistent estimators of $v(z|b)$ and $v(sq|z,b)$ respectively. This assumption is satisfied if either a frequency estimator, a CTW estimator, or some other consistent estimator is used for $\hat{v}(z|b)$ and $\hat{v}(sq|z,b)$.

We can now move on to the main theoretical contribution of this work.

Theorem 5.7. *Given an m-horizon, finite state space, finite action space, time-homogenous MDP $\mathcal{M} := (\widehat{\mathcal{S}}, \widehat{\mathcal{A}}, \vartheta)$ and a stationary policy π that gives rise to a an (IR+EA+PR) HMC, then for any state $sq \in \widehat{\mathcal{S}}$ and action $b \in \widehat{\mathcal{A}}$ such that $\tau(D_\rho(qb|s)) = a$, we have*

$$\hat{Q}^\pi_{\vartheta,n}(sq, b) \overset{a.s.}{\to} Q^\pi_\mu(s, a) \tag{5.7}$$

provided $\hat{v}(z|b)$ and $\hat{v}(sq|z,b)$ are consistent estimators of $v(z|b)$ and $v(sq|z,b)$ respectively. Furthermore, $\hat{Q}^\pi_{\vartheta,n}(sq,b)$ has the convergence rate of $w_n^{z,b}(sq)$.

Proof. From the assumptions we have that $w_n^{z,b}(sq)$ is a consistent estimator. Let the convergence rate of $w_n^{z,b}(sq)$ be $O_{\mathbb{P}}(f(n))$.

$$\begin{aligned}
|\hat{Q}^\pi_{\vartheta,n}(sq,b) - Q^\pi_\mu(s,a)| &\overset{(a)}{=} |\hat{Q}^\pi_{\vartheta,n}(sq,b) - Q^\pi_\vartheta(sq,b)| \\
&\overset{(b)}{=} \left| \sum_{z\in\mathcal{Z}} z\, w_n^{z,b}(sq) - \sum_{z\in\mathcal{Z}} z\, v(z|sq,b) \right| \\
&\overset{(c)}{\leq} \sum_{z\in\mathcal{Z}} z \left| w_n^{z,b}(sq) - v(z|sq,b) \right| \\
&\overset{(d)}{=} O_{\mathbb{P}}(f(n))
\end{aligned}$$

(a) comes from Theorem 5.2. (b) comes from the definition of $\hat{Q}^\pi_{\vartheta,n}$ and Equation 5.5. (c) is the triangle inequality. (d) comes from the convergence rate of $w_n^{z,b}(sq)$, and the fact that $\mathcal{Z}$ is finite. □

This result states that, under reasonable assumptions, the approximation of our binary action action-value function, $\hat{Q}^\pi_{\vartheta,n}$, converges to the (non-binary action) action-value function $Q^\pi_\mu(s,a)$, and that this convergence is fast (assuming the convergence of $w_n^{z,b}$ is fast).

With this result, we have shown that Compress and Converse is a theoretically viable method for policy evaluation.

Consistency is immortal

A immediate consequence of Proposition 4.3 and Theorem 5.7 is that for a given ρ and π, if $\hat{Q}^{\pi}_{\vartheta,n}$ is a consistent estimator of Q^{π}_{ϑ}, then if the action model ρ is pre-trained on additional data, then the external policy is still a consistent estimator.

Corollary 5.8. *If $\hat{v}(z|b)$ and $\hat{v}(sq|z,b)$ are consistent estimators, then the estimator $\hat{Q}^{\pi}_{\vartheta,n}(sq,b)$, with some initial samples, is consistent.*

Proof. Consequence of Theorem 5.7 and Proposition 4.3. □

In practice pre-training the estimators is often done to speed up convergence times, and the above result shows that it does not alter the convergence properties of $\hat{Q}^{\pi}_{\vartheta,n}$.

5.5 Discussion

Imagining the future. Due to the nature of this setup we will be able to utilise the action model not just for deriving current actions, but as a method to efficiently predict future actions. By sampling from the action model, we have a fast method for performing the rollout component of a search algorithm. However, using the action model incorrectly can lead to auto-suggestive delusions (Ortega, Kunesch et al., 2021). Auto-suggestive delusion take place when the model uses it's own generated actions as evidence.

Choices of state space representation. In our setup we use the augmented state space $\widehat{\mathcal{S}} := \mathcal{S} \times \mathbb{B}^{<n}$, where $\mathcal{S}$ is often the space of sequences of characters with bounded length, or words. For implementation purposes, we have a choice of how we want to represent this state space; we could represent $\mathcal{S}$ as it, or alternatively we could use a binary encoding of $\mathcal{S}$, meaning that $\widehat{\mathcal{S}}$ becomes the space of (bounded-length) binary strings. If this is done, then we may also need some way to tell when $\mathcal{S}$ ends and when $\mathbb{B}^{<n}$ begins.

On the limitations of language models without online RL. Given the power and flexibility of Large Language Models (LLM), and access to many recorded expert trajectories in RL domains, it is natural to consider distilling such expert knowledge into a pre-trained LLM which represents a policy. Some recent work has even advocated that RL can be viewed as one large sequence prediction problem (Janner et al., 2021). While this may work in some very simple environments in general it will not work well. This is for a number of reasons: Firstly, since the language model is pre-trained it does not learn during interactions with the environment (the very point/goal of reinforcement learning), so unless it starts as the optimal policy it will perform sub-optimally forever; secondly, it involves no planning and does not use the reward from the environment at all.

Legal actions. In many games and environments, the set of possible actions and the set of legal actions (at a given state) can be vastly different. While we will often want the action model to learn which actions are legal and illegal, this can sometimes be difficult. In these

cases, a possible approach would be to enumerate legal actions and re-normalise. This would however require "outside" knowledge of the mechanics of the environment/game.

Determining probabilities from codes. While translating probabilities to codes through the arithmetic decoder can easily be done, there is no efficient method for translating given codes into a probability distribution. This is because by the nature of arithmetic decoding a given code only tells us one part of the probability mass (essentially a lower bound), and the probability mass may "spill over" to adjacent intervals.

5.6 Future Work

A potential direction of this research is to ignore the compress part of Compress and Converse and use a different policy/policy evaluation method. This different policy/policy evaluation method will have to overcome or ideally completely avoid the common problems of the binarisation of actions, such as increased planning horizon.

There are some additional extensions that could be made to Compress and Converse from a theoretical perspective. These include: considering non-stationary policies, extend from Markov to k-Markov, dropping aperiodicity assumptions and using time-averages, generalising from ergodic Markov processes to general ergodic processes, and finding the hidden multiplicative/additive constants, which will depend on mixing time of HMC (S_t, A_t).

5.7 Conclusion

In this chapter, we evaluated the binary action setup of information-theoretic actuation by applying it to a policy evaluation technique that can take advantage of it. We have shown that this approach is valid by demonstrating that our approximations of the action-value function in the binary setup will converge to the true action-value.

Chapter 6

On Reward Binarisation and Bayesian Agents

This chapter has been presented as

E. Catt, J. Veness et al. (2022b). 'On Reward Binarisation and Bayesian Agents'. In: *European Workshop on Reinforcement Learning 15*

Abstract

Reward binarisation is a common heuristically applied technique which can potentially simplify a given reinforcement learning problem. However this procedure done without care can modify the original problem, or throw away essential information. In this chapter we study a number of natural forms of reward binarisation, and characterise their effects in terms of problem expressitivity. We show positive results for MDPs, POMDPs, and k-order MDPs and a negative result for general history based reinforcement learning agents. Furthermore we show that binary Bayesian reinforcement learning agents enjoy convergence properties similar to their non-binarised counterparts.

6.1 Introduction

Various kinds of reward shaping are common practice for many reinforcement learning practitioners. This chapter considers some theoretical aspects of an extreme class of shaping methods, known as reward binarisation, whereby each reward received by the agent is mapped via some particular mechanism to either a 0 or a 1. Indeed, the origins of reinforcement learning date back to presence (one) or lack of (zero) reward with operant conditioning (Breland and Breland, 1961; Pavlov, 1927). Perhaps surprisingly a lot can be said in this case, especially in terms of representation power and ability of an agent to learn provided an appropriate binarisation scheme is chosen.

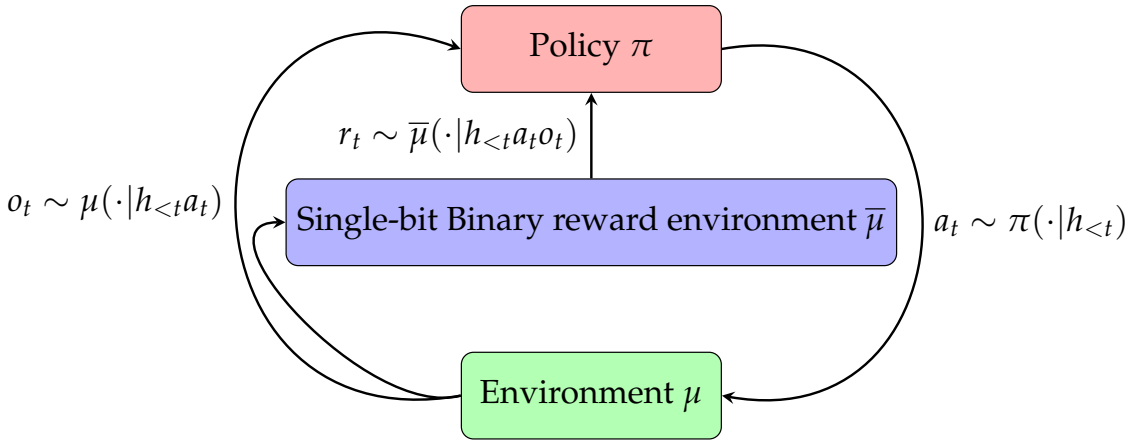

Figure 6.1: Agent-environment loop with single-bit reward.

While there are many possible binarisation schemes, there are a couple of natural candidates. In this work we consider two approaches, one which binarises the reward to a single bit, and another scheme which iteratively presents the reward to the agent one bit at a time over multiple time steps. We show that in the case of Markov Decision Processes, binary rewards are a sufficient reward signal to exactly capture the value function; this extends to k-Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs) as well. We also provide a negative result, which shows that in a more general history based setup, binarisation cannot hope to preserve the value function. Figure 6.2 shows the resultant reward binarisations we consider.

Binarisation Type	Equivalent of aor	Notes
Single bit	$ao0 \vee ao1$	Possibly lossy
Multi-bit	$aor_1a'or_2 \ldots a'or_d$	a' is any action

Figure 6.2: Reward binarisation types

The second part of our chapter deals with Bayesian learning in the presence of reward binarisation. Here we provide a number of technical results that justify the use of Bayesian

learning in such situations. More precisely, we show that a Bayesian mixture of the set of binarised environments is a dominant semimeasure, and so is the binarised form of the original Bayesian mixture. Furthermore, using similar techniques, we show that Bayesian learning on top of a recently proposed action binarisation scheme (Majeed and Hutter, 2021) is also justified theoretically.

6.2 Background

In this chapter, as well as Chapter 7 and Chapter 8 we use a general form of reinforcement learning notation which will allow our results to cover multiple reinforcement learning setups with a unified notation. Here we will expand on the notation presented in Chapter 2.

Setup. We will use the general reinforcement learning framework of (Catt, Quarel et al., 2022; Hutter, 2005; Leike, 2016; Majeed and Hutter, 2021). Let $\mathcal{A}$ be the finite action set, $\mathcal{O}$ be the finite observation set, $\{0,1\} \subset \mathcal{R} \subset [0,1]$ be the finite (non-binary) reward set and $\mathcal{H} := (\mathcal{A} \times \mathcal{O} \times \mathcal{R})^* = \cup_{t=0}^{\infty} (\mathcal{A} \times \mathcal{O} \times \mathcal{R})^t$ be the set of finite length interaction histories (empty history in the $t = 0$ case). In this setup agents, denoted by $\pi : \mathcal{H} \to \Delta \mathcal{A}$, take actions $a \in \mathcal{A}$ and receive observations $o \in \mathcal{O}$ and rewards $r \in \mathcal{R}$ from the environment, denoted by $\mu : \mathcal{H} \times \mathcal{A} \to \Delta(\mathcal{O} \times \mathcal{R})$. The history of interactions up to time t, $a_1 o_1 r_1 \ldots a_{t-1} o_{t-1} r_{t-1}$ is denoted by $h_{<t}$, and $h_{1:t} := h_{<t} a_t o_t r_t$. We denote the empty string by ϵ.

Value Functions. The goal of the agent in this setup is to maximise the expected future reward from the environment; this expected future reward is called the value function

$$V_\mu^{\pi,m}(h_{<t}) := \mathbb{E}_\mu^\pi \left[\sum_{i=t}^{m} \gamma(i) r_i | h_{<t} \right],$$

where γ is the (often geometric) discount function and m is the maximum horizon. The optimal value is defined as $V_\nu^{*,m}(h_{<t}) := \sup_\pi V_\nu^{\pi,m}(h_{<t})$, the optimal policy with respect to the value is defined as $\pi_\nu^{*,m}(h_{<t}) \in \arg\max_\pi V_\nu^{\pi,m}(h_{<t})$. In the case when $m = \infty$ we omit the m, that is, $V_\nu^\pi(h_{<t}) := V_\nu^{\pi,\infty}(h_{<t})$. We will also drop the history argument when it is an empty history, $V_\nu^\pi := V_\nu^\pi(\epsilon)$.

Unknown Environments. If the agent does not initially know which environment it is in, a model based approach would be to consider a class of possible environments. Let $\mathcal{M}$ denote the class of possible environments; in this chapter we allow for both finite $\mathcal{M}$ and countable $\mathcal{M}$, with special attention being given to the case when $\mathcal{M}$ is the set of enumerable semimeasures. A environment ν is a semimeasure if $\sum_{o_t r_t} \nu(o_t r_t | h_{<t} a_t) \leq 1$ for all $h_{<t}, a_t$ and a (proper) measure if equality holds instead.

Bayesian Reinforcement Learning. The Bayesian-optimal agent AIXI considers a Bayesian mixture of the class of possible environments and acts optimally with respect to the Bayesian

mixture (Hutter, 2005, 2006; Leike, 2016). We define the Bayesian mixture over the class of environments as follows:

$$\xi_{\mathcal{M}}(h) := \sum_{\nu \in \mathcal{M}} w(\nu)\nu(h)$$

where $w(\nu) > 0$ is the prior probability of the environment ν.

One of the key properties of the Bayesian mixture $\xi_{\mathcal{M}}$ is it's dominance (Rathmanner and Hutter, 2011). This means that for all $\nu \in \mathcal{M}$ and for all x we have that $\xi_{\mathcal{M}}(x) \geq c\nu(x)$. This property is important as it can be used to show that if μ is the true environment then ξ learns to correctly predict μ, specifically, $\xi_{\mathcal{M}} \to \mu$ with μ-probability 1.

MDPs/POMDPs. We will additionally consider the simplified Markov Decision process setting of reinforcement learning wherein the agent and environment depend only on the most recent observation instead of the whole history. Lastly we will will consider the partially observable markov decision processes (POMDP) (Kaelbling, Littman and Cassandra, 1998; Kaelbling, Littman and Moore, 1996) class of environments. A POMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mu_T, R, \mathcal{O}, \mu_O, \gamma \rangle$ where $\mathcal{S}$ is a finite set of hidden states, $\mathcal{A}$ is a set of actions, μ_T is the transition probability of the hidden states, R is the reward function, $\mathcal{O}$ is the set of possible observations, μ_O is the observation probability given a state, and γ is the discount factor. POMDPs can be represented in general reinforcement learning with the following definition from Majeed (2021). An environment μ is a POMDP if there exists a finite state space $\mathcal{S}$, emission process μ_O, underlying MDP μ_T, and true occupancy probability χ such that

$$\mu(o'r'|ha) = \sum_{s' \in \mathcal{S}} \mu_O(o'r'|s') \sum_{s \in \mathcal{S}} \mu_T(s'|sa)\chi(s|h).$$

6.3 Binary Reward (Single-bit) Formulation

When considering the binarisation of the reward we will separately consider a specific case of when the rewards are binarised to a single bit, as opposed to the reward being binarised into several bits. Unless the original reward space $\mathcal{R}$ has a cardinality of 2, this will be a lossy binarisation in the sense that information (may) will be lost during this binarisation. It turns out however, that despite this being a lossy binarisation, we are still able to show that in some cases it is sufficient.

In comparison to multi-bit binarisation, if we use single-bit binarisation we will not need to artificially add timesteps, this can be very important for methods which depend on the horizon, which becomes modified in the action binarisation and multi-bit reward binarisation cases.

Let $\overline{\mathcal{M}} := \{\mu \in \mathcal{M} | \mu \text{ only outputs reward 0 and 1}\}$ be the class of environments which output only binary rewards. Since we are considering the class of enumerable semimeasures this just means that the probability that the environment outputs a reward which isn't in $\{0, 1\}$ is 0.

In lieu of a specific binarisation scheme, we will use $\langle r \rangle$ to represent a binarisation of the reward to a single bit and $\langle h_{<t} \rangle = a_1 o_1 \langle r_1 \rangle \dots a_{t-1} o_{t-1} \langle r_{t-1} \rangle$ to represent a binarisation of the reward parts of the history. This is an intended feature of this setup as the results hold for any choice of reward binarisation.

Similarly to Definition 6.17, we will define a binarisation of an environment μ to one which uses only a binary reward.

Definition 6.1 (General Single-bit reward binarisation)**.** *Given μ, we define the environment which outputs only binary reward $\overline{\mu} : \mathcal{H} \times \mathcal{A} \to \Delta(\mathcal{O} \times \{0,1\})$ in two parts:*

$$\overline{\mu}(o_t 1 | h_{<t} a_t) := \sum_{r_t \in \mathcal{R}} \mu(o_t r_t | h_{<t} a_t) r_t$$
$$\overline{\mu}(o_t 0 | h_{<t} a_t) = \sum_{r_t \in \mathcal{R}} \mu(o_t r_t | h_{<t} a_t)(1 - r_t)$$

and $\overline{\mu}(o_t r | h_{<t} a_t) = 0$ if $r \notin \{0,1\}$.

The reason we do $\sum_{r_t \in \mathcal{R}} \mu(o_t r_t | h_{<t} a_t)(1 - r_t)$ instead of $1 - \overline{\mu}(o_t 1 | h_{<t} a_t)$ is that there may be a semimeasure gap in ρ; that is, $\left(\sum_{r_t \in \mathcal{R}} \mu(o_t r_t | h_{<t} a_t)\right)$ may not equal 1.

We will now show that the transformation in Definition 6.1 is a linear transformation over environments. This will be important for later results.

Lemma 6.2. *If an enumerable semimeasure ρ can be written as the linear combination of two enumerable semimeasures, i.e. for $x, y \in \mathbb{R}$ if $\rho = x\mu_1 + y\mu_2$, then $\overline{\rho} = x\overline{\mu}_1 + y\overline{\mu}_2$. That is, the operator $\bar{\cdot}$ is linear.*

Proof. For reward 1,

$$\begin{aligned}
\overline{\rho}(o1|ha) &= \sum_{r \in \mathcal{R}} \rho(or|ha) \cdot r \\
&= \sum_{r \in \mathcal{R}} (x\mu_1(or|ha) + y\mu_2(or|ha)) \cdot r \\
&= \sum_{r \in \mathcal{R}} (x\mu_1(or|ha)r + y\mu_2(or|ha)r) \\
&= x \sum_{r \in \mathcal{R}} \mu_1(or|ha)r + y \sum_{r \in \mathcal{R}} \mu_2(or|ha)r \\
&= x\overline{\mu}_1(o1|ha) + y\overline{\mu}_2(o1|ha).
\end{aligned}$$

For reward 0,

$$\begin{aligned}
\overline{\rho}(o0|ha) &= \sum_{r\in\mathcal{R}} \rho(or|ha)\cdot(1-r) \\
&= \sum_{r\in\mathcal{R}} (x\mu_1(or|ha) + y\mu_2(or|ha))\cdot(1-r) \\
&= \sum_{r\in\mathcal{R}} (x\mu_1(or|ha)(1-r) + y\mu_2(or|ha)(1-r)) \\
&= x\sum_{r\in\mathcal{R}} \mu_1(or|ha)(1-r) + y\sum_{r\in\mathcal{R}} \mu_2(or|ha)(1-r) \\
&= x\overline{\mu}_1(o0|ha) + y\overline{\mu}_2(o0|ha).
\end{aligned}$$

Therefore we have $\overline{\rho}(or|ha) = x\overline{\mu}_1(or|ha) + y\overline{\mu}_2(or|ha)$. □

6.4 Value Function Results

In this section we will present several results regarding the reward binarisation and the corresponding value functions.

We will take a small diversion here to present a lemma which will be useful for a later proof. Effectively this lemma says that given a set of history-real number pairs, there exists an environment such that the value function in that environment on each of those histories is that real number.

Formally, let $\mathcal{Q} \subset \mathcal{H}\times(0,1]$ be such that for all $h\in\mathcal{H}$ there exists a unique $q\in(0,1]$ such that $(h,q)\in\mathcal{Q}$.

Lemma 6.3. *For a given $\mathcal{Q}$, there exists a binary reward environment μ' such that for all $(h,q)\in\mathcal{Q}$ and for all policies π we have $V_{\mu'}^{\pi,m}(h) = q$.*

Proof. We will prove a more general result and show that there are infinitely many choices of μ that satisfy $V_{\mu'}^{\pi,m}(h) = q$ for an arbitrary history h. We do this by showing it is true at the maximum horizon m, then work backwards.

Let $q(h)$ denote the q such that $(h,q)\in\mathcal{Q}$.

$$\begin{aligned}
\mu'(o_t1|h_{<t}a_t) &:= \begin{cases} \frac{q(h_{<m})}{|\mathcal{O}|} & \text{if } t = m \\ \frac{\frac{q(h_{<t})}{|\mathcal{O}|} - \gamma q(h_{<t}a_to_t0)}{1+\gamma q(h_{<t}a_to_t1) - \gamma q(h_{<t}a_to_t0)} & \text{if } t < m \end{cases} \\
\mu'(o_t0|h_{<t}a_t) &:= 1 - \mu'(o_t1|h_{<t}a_t).
\end{aligned}$$

In the case that $t = m$,

$$\begin{aligned} V_{\mu'}^{\pi,m}(h_{<m}) &= \sum_{a_t \in \mathcal{A}} \pi(a_t|h_{<t}) \sum_{o_m r_m} \mu'(o_m r_m | h_{<m} a_m)(r_m) \\ &= \sum_{a_t \in \mathcal{A}} \pi(a_t|h_{<t}) q(h_{<m}) \\ &= q(h_{<m}) \sum_{a_t \in \mathcal{A}} \pi(a_t|h_{<t}) \\ &= q(h_{<m}). \end{aligned}$$

In the second case that $t < m$, we can work backwards from $t = m$.

$$\begin{aligned} V_{\mu'}^{\pi,m}(h_{<t}) &= \sum_{a_t \in \mathcal{A}} \pi(a_t|h_{<t}) \sum_{o_t r_t} \mu'(o_t r_t | h_{<t} a_t)(r_t + \gamma V_{\mu'}^{\pi,m}(h_{<t} a_t o_t r_t)) \\ &= \sum_{a_t \in \mathcal{A}} \pi(a_t|h_{<t}) \sum_{o_t r_t} \mu'(o_t r_t | h_{<t} a_t)(r_t + \gamma q(h_{<t} a_t o_t r_t)) \\ &= \sum_{a_t \in \mathcal{A}} \pi(a_t|h_{<t}) \sum_{o_t} \big((1 - \mu'(o_t 1 | h_{<t} a_t))(\gamma q(h_{<t} a_t o_t 0)) + \mu'(o_t 1 | h_{<t} a_t)(1 + \gamma q(h_{<t} a_t o_t 1))\big) \\ &= \sum_{a_t \in \mathcal{A}} \pi(a_t|h_{<t}) \sum_{o_t} \big(\gamma q(h_{<t} a_t o_t 0) + \mu'(o_t 1 | h_{<t} a_t)(1 + \gamma q(h_{<t} a_t o_t 1) - \gamma q(h_{<t} a_t o_t 0))\big) \\ &= \sum_{a_t \in \mathcal{A}} \pi(a_t|h_{<t}) \sum_{o_t} \left(\gamma q(h_{<t} a_t o_t 0) + \frac{q(h_{<t})}{|\mathcal{O}|} - \gamma q(h_{<t} a_t o_t 0))\right) \\ &= \sum_{a_t \in \mathcal{A}} \pi(a_t|h_{<t}) q(h_{<t}) \\ &= q(h_{<t}). \end{aligned}$$

□

6.4.1 Impossibility of exact representation in general setting

We will now show that for any choice of reward-binarising function $\langle\cdot\rangle$, the value function cannot be exactly represented with a binary reward environment.

Theorem 6.4. *Given a reward-binarising function $\langle\cdot\rangle : \mathcal{R} \to \{0,1\}$, there does not exist an environment reward binarising function $\bar{\cdot} : \mathcal{M} \to \overline{\mathcal{M}}$ such that for all $\mu \in \mathcal{M}$, $h_{<t} \in \mathcal{H}$ and π we have*

$$V_{\bar{\mu}}^{\pi}(\langle h_{<t} \rangle) = V_{\mu}^{\pi}(h_{<t}).$$

Proof. Let $\mathcal{R} = \{0, \frac{1}{2}, 1\}$ and $\mathcal{O} = \{o\}$. Let μ be such that for $t > 1$

$$\begin{aligned}
&\text{If } r_1 = 0 \\
&\mu(o0|h_{<t}a_t) = 0,\ \mu(o\frac{1}{2}|h_{<t}a_t) = 0,\ \mu(o1|h_{<t}a_t) = 1. \\
&\text{If } r_1 = \frac{1}{2} \\
&\mu(o0|h_{<t}a_t) = \frac{1}{3},\ \mu(o\frac{1}{2}|h_{<t}a_t) = 0,\ \mu(o1|h_{<t}a_t) = \frac{2}{3}. \\
&\text{If } r_1 = 1 \\
&\mu(o0|h_{<t}a_t) = \frac{2}{3},\ \mu(o\frac{1}{2}|h_{<t}a_t) = 0,\ \mu(o1|h_{<t}a_t) = \frac{1}{3}.
\end{aligned}$$

This means that

$$\begin{aligned}
V^\pi_\mu(ao0h) &= 1 + \gamma + \gamma^2 \ldots = \frac{1}{1-\gamma} \\
V^\pi_\mu(ao\frac{1}{2}h) &= \frac{2}{3}(1 + \gamma + \gamma^2 \ldots) = \frac{2}{3(1-\gamma)} \\
V^\pi_\mu(ao1h) &= \frac{1}{3}(1 + \gamma + \gamma^2 \ldots) = \frac{1}{3(1-\gamma)}
\end{aligned}$$

Since $\langle\cdot\rangle$ binarises the reward component of the history, (at least) two of $ao0h, ao\frac{1}{2}h, ao1h$ will map to the same binarised history, then by a pigeon hole argument we will have two of the above value functions needing to be equal, however that cannot occur, so the equality

$$V^\pi_{\overline{\mu}}(\langle h_{<t}\rangle) = V^\pi_\mu(h_{<t})$$

cannot be satisfied. □

While this result holds for any choice of binarisation of the environment (not limited to just Definition 6.1), it is a result about exact representation of the value function. It may still be possible that with binary rewards we can get within ε of the value function, like Theorem 6.20.

6.4.2 Possibility of exact representation

We have just shown that it is impossible to achieve exact representation of the value function for a general environment. We will now move on to showing that for a large interesting class of environments, it is possible to achieve exact representation of the value function. We will start by defining this class of environment, and then showing that it contains several environment classes of note, and lastly proving that any environment in this class can be represented by a corresponding binary environment.

Definition 6.5. *Let $\mathcal{M}^*$ be the set of environments for which the transitions do not depend on the reward components of the history. Formally:*

$$\mathcal{M}^* := \{\mu \in \mathcal{M} \mid \forall h_{<t}, a_t, o_t, r_t.\ \mu(o_t r_t | h_{<t} a_t) = \mu(o_t r_t | a_1 o_1 \dots a_{t-1} o_{t-1} a_t)\}. \tag{6.1}$$

The history independence of reward r is not a restriction since rewards can be encoded in the observations o, but then they don't get binarised, which presents the earlier counter-example.

We can now define the environment reward binarisation we will be using to convert environment in $\mathcal{M}^*$ to binary reward environments.

Definition 6.6 (Single-bit reward binarisation for $\mathcal{M}^*$)**.** *Given $\mu \in \mathcal{M}^*$, we define the environment which outputs only binary reward $\overline{\mu} : \mathcal{H} \times \mathcal{A} \to \Delta(\mathcal{O} \times \{0,1\})$ in two parts:*

$$\begin{aligned}
\overline{\mu}(o_t 1 | h_{<t} a_t) &:= \left(\sum_{r_t \in \mathcal{R}} \mu(o_t r_t | h_{<t} a_t) \left(r_t + \gamma V_\mu^{\pi,m}(h_{<t} a_t o_t r_t) \right) \right) \\
&\quad - \gamma V_\mu^{\pi,m}(h_{<t} a_t o_t) \\
\overline{\mu}(o_t 0 | h_{<t} a_t) &:= 1 - \overline{\mu}(o_t 1 | h_{<t} a_t).
\end{aligned}$$

Furthermore, we say that a policy π is reward history independent if $\pi(a_t | h_{<t}) = \pi(a_t | a_1 o_1 \dots a_{t-1} o_{t-1})$ for all $h_{<t}$ and a_t. As we are interested in policies which learn to perform well, and performing well means achieving high reward over time, it is natural that policies *should* depend on the rewards in this history, however most RL theory assumes this is not the case as it makes the analysis easier. We now state our main result regarding the possibility of representation for $\mu \in \mathcal{M}^*$.

Theorem 6.7. *If π is reward history independent then for all $\mu \in \mathcal{M}^*$, all $h_{<t}$ and all choices of history binarising function $\langle \cdot \rangle$ we have*

$$V_{\overline{\mu}}^{\pi,m}(\langle h_{<t} \rangle) = V_\mu^{\pi,m}(h_{<t}).$$

Before we show that environments in $\mathcal{M}^*$ can be represented in by binary reward environments we will need a small lemma about $\mathcal{M}^*$.

Lemma 6.8. *If π is reward history independent, that is, $\pi(a|h) = \pi(a|a_1 o_1 \dots a_{t-1} o_{t-1})$ then for all $\mu \in \mathcal{M}^*$ and $h_{<t}$ we have that*

$$V_\mu^\pi(\langle h_{<t} \rangle) = V_\mu^\pi(h_{<t})$$

and

$$V_\mu^\pi(h_{<t}) = V_\mu^\pi(h_{<t-1} a_{t-1} o_{t-1})$$

Proof. This come from the fact that the distributions in the expectation do not depend on the reward components of the history. □

Proof of Theorem 6.7.

Proof. Proof sketch: expand the definitions of the value function and $\overline{\mu}$ and apply Lemma 6.8.

In the $t = m$ case:

$$\begin{aligned}
V_{\overline{\mu}}^{\pi,m}(\langle h_{<m}\rangle) &= \sum_{a_m \in \mathcal{A}} \pi(a_m|\langle h_{<m}\rangle) \sum_{o_m r_m} \overline{\mu}(o_m r_m|\langle h_{<m}\rangle a_m)(r_m) \\
&= \sum_{a_m \in \mathcal{A}} \pi(a_m|\langle h_{<m}\rangle) \sum_{o_m} \overline{\mu}(o_m 1|\langle h_{<m}\rangle a_m) \\
&= \sum_{a_m \in \mathcal{A}} \pi(a_m|\langle h_{<m}\rangle) \sum_{o_m} \sum_{r_m \in \mathcal{R}} \mu(o_m r_m|\langle h_{<m}\rangle a_m) r_m \\
&= V_{\mu}^{\pi,m}(\langle h_{<m}\rangle) \\
&= V_{\mu}^{\pi,m}(h_{<m})
\end{aligned}$$

In the second case that $t < m$, we can work backwards from $t = m$.

$$\begin{aligned}
V_{\overline{\mu}}^{\pi,m}(\langle h_{<t}\rangle) &\overset{(a)}{=} \sum_{a_t \in \mathcal{A}} \pi(a_t|\langle h_{<t}\rangle) \sum_{o_t r_t} \overline{\mu}(o_t r_t|\langle h_{<t}\rangle a_t)(r_t + \gamma V_{\overline{\mu}}^{\pi,m}(\langle h_{<t}\rangle a_t o_t r_t)) \\
&\overset{(b)}{=} \sum_{a_t \in \mathcal{A}} \pi(a_t|\langle h_{<t}\rangle) \sum_{o_t r_t} \overline{\mu}(o_t r_t|\langle h_{<t}\rangle a_t)(r_t + \gamma V_{\overline{\mu}}^{\pi,m}(\langle h_{<t}\rangle a_t o_t)) \\
&\overset{(c)}{=} \sum_{a_t \in \mathcal{A}} \pi(a_t|\langle h_{<t}\rangle) \\
&\quad \cdot \sum_{o_t} \Big(\overline{\mu}(o_t 0|\langle h_{<t}\rangle a_t)(\gamma V_{\overline{\mu}}^{\pi,m}(\langle h_{<t}\rangle a_t o_t)) + \overline{\mu}(o_t 1|\langle h_{<t}\rangle a_t)(1 + \gamma V_{\overline{\mu}}^{\pi,m}(\langle h_{<t}\rangle a_t o_t)) \Big) \\
&\overset{(d)}{=} \sum_{a_t \in \mathcal{A}} \pi(a_t|\langle h_{<t}\rangle) \\
&\quad \cdot \sum_{o_t} \Big((1 - \overline{\mu}(o_t 1|\langle h_{<t}\rangle a_t))(\gamma V_{\overline{\mu}}^{\pi,m}(\langle h_{<t}\rangle a_t o_t)) + \overline{\mu}(o_t 1|\langle h_{<t}\rangle a_t)(1 + \gamma V_{\overline{\mu}}^{\pi,m}(\langle h_{<t}\rangle a_t o_t)) \Big) \\
&\overset{(e)}{=} \sum_{a_t \in \mathcal{A}} \pi(a_t|\langle h_{<t}\rangle) \sum_{o_t} \Big(\overline{\mu}(o_t 1|\langle h_{<t}\rangle a_t) + \gamma V_{\overline{\mu}}^{\pi,m}(\langle h_{<t}\rangle a_t o_t) \Big) \\
&\overset{(f)}{=} \sum_{a_t \in \mathcal{A}} \pi(a_t|\langle h_{<t}\rangle) \sum_{o_t} \Big(\overline{\mu}(o_t 1|\langle h_{<t}\rangle a_t) + \gamma V_{\mu}^{\pi,m}(\langle h_{<t}\rangle a_t o_t) \Big) \\
&\overset{(g)}{=} \sum_{a_t \in \mathcal{A}} \pi(a_t|\langle h_{<t}\rangle) \sum_{o_t} \left(\sum_{r_t \in \mathcal{R}} \mu(o_t r_t|\langle h_{<t}\rangle a_t) \left(r_t + \gamma V_{\mu}^{\pi,m}(\langle h_{<t}\rangle a_t o_t r_t) \right) \right) \\
&\overset{(h)}{=} V_{\mu}^{\pi,m}(\langle h_{<t}\rangle) \\
&\overset{(i)}{=} V_{\mu}^{\pi,m}(h_{<t})
\end{aligned}$$

(a) comes from the Bellman equation of V. (b) comes from Lemma 6.8. (c) involves expanding the reward sum for the two rewards of 0 and 1. (d) and (e) are algebra. (f) comes from the $t+1$ case working backwards. (g) comes from Definition 6.6. (h) is the Bellman equation of V. (i) is Lemma 6.8. This completes the proof. □

An immediate consequence of this is that the optimal agent for $\overline{\mu}$ is optimal for the environment μ.

Corollary 6.9. *If $\mu \in \mathcal{M}^*$ then $\pi^*_{\overline{\mu}} \in \arg\max_\pi V^\pi_\mu$.*

We will now show that some well studied environment classes in reinforcement learning are subclasses of $\mathcal{M}^*$.

Proposition 6.10. $\mathcal{M}_{MDP}, \mathcal{M}_{kMDP}, \mathcal{M}_{POMDP} \subseteq \mathcal{M}^*$.

These follow trivially from the definition of $\mathcal{M}^*$, as in MDPs, k-MDPs, and POMDPs the transitions probabilities only depend on the previous observations and actions, and not the rewards.

6.5 The Case of Bayesian RL

As discussed earlier, the Bayesian solution to the general reinforcement learning problem involves considering a Bayesian mixture over the class of environments in the case where μ is unknown. In this case it is well known that predictive distribution of the mixture will converge to the true environment if the true environment is contained within the mixture class. In this section we show that reward binarisation preserves this property. This result covers the case of finite mixtures, but also generalises to idealised Bayesian agents such as AIXI (Hutter, 2005) whose mixture class contains countably many environments.

To show this, it is sufficient to show that the mixture over binarised environments $\xi_{\overline{\mathcal{M}}}$ and the binarised form of the mixture $\overline{\xi}$ are a dominant semimeasures in $\mathcal{M}$.

Theorem 6.11. *For all $\mu \in \mathcal{M}$, there exists a $c > 0$ such that for all $h_{<t} \in \mathcal{H}$ we have*

$$\xi_{\overline{\mathcal{M}}}(\langle h_{<t}\rangle) > c\mu(h_{<t})$$

That is, $\xi_{\overline{\mathcal{M}}}$ is a dominant semimeasure in $\mathcal{M}$.

Proof. Proof idea: We show that for every possible reward string there exists a enumerable binary environment such that that binary environment with that reward string correspond to the chosen μ.

We can consider a binary reward environment for each possible $r_{1:t} \in \mathcal{R}^{t-1}$.

Let $\mu_{r,r'}$ be defined on the observation space as

$$\mu_{r,r'}(o_k|\langle h_{<k}\rangle a_k) := \begin{cases} \mu(o_k|h'_{<k}a_k) & \text{if } \langle r'\rangle = \langle r_{<k}\rangle \\ \frac{1}{|\mathcal{O}|} & \text{otherwise} \end{cases}$$

and defined on the reward space as

$$\mu_{r,r'}(\langle r_k\rangle|\langle h_{<k}\rangle a_k o_k) = \begin{cases} \mu(r|h'_{<k}a_k o_k) & \text{if } \langle r'\rangle\langle r\rangle = \langle r_{<k}\rangle\langle r_k\rangle \\ 1-\mu(r|h'_{<k}a_k o_k) & \text{if } \langle r'\rangle = \langle r_{<k}\rangle \wedge \langle r\rangle \neq \langle r_k\rangle \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

where $h'_{<k}$ is $h_{<k}$ with the reward sequence replace by r'. Since μ is enumerable, $\mu_{r,r'}$ is also enumerable for each r, r'. Note that r' is a string of rewards and r is a single reward.

Therefore we have

$$\begin{aligned}
\xi_{\overline{\mathcal{M}}}(\langle h_{<t}\rangle) &= \sum_{\nu\in\overline{\mathcal{M}}} w(\nu)\nu(\langle h_{<t}\rangle) \\
&> \sum_{r\in\mathcal{R},r'\in\mathcal{R}^*} w(\mu_{r,r'})\mu_{r,r'}(\langle h_{<t}\rangle) \\
&= \sum_{r\in\mathcal{R},r'\in\mathcal{R}^*} w(\mu_{r,r'})\prod_{k=1}^{t-1}\mu_{r,r'}(o_k\langle r_k\rangle|\langle h_{<k}\rangle a_k) \\
&= \sum_{r\in\mathcal{R},r'\in\mathcal{R}^*} w(\mu_{r,r'}) \\
&\cdot\prod_{k=1}^{t-1}\mu_{r,r'}(o_k|\langle h_{<k}\rangle a_k)\mu_{r,r'}(\langle r_k\rangle|\langle h_{<k}\rangle a_k o_k) \\
&> w(\mu_{r_{t-1},r_{1:t-2}}) \\
&\cdot\prod_{k=1}^{t-1}\mu_{r_{t-1},r_{1:t-2}}(o_k|\langle h_{<k}\rangle a_k)\mu_{r_{t-1},r_{1:t-2}}(\langle r_k\rangle|\langle h_{<k}\rangle a_k o_k) \\
&= w(\mu_{r_{t-1},r_{1:t-2}})\prod_{k=1}^{t-1}\mu(o_k|h_{<k}a_k)\mu(r_k|h_{<k}a_k o_k) \\
&= w(\mu_{r_{t-1},r_{1:t-2}})\mu(h_{<t})
\end{aligned}$$

where $\mathcal{R}^* = \cup_{t=0}^{\infty} R^t$.

If we choose $c = \sup\{w(\mu_{r,r'}) \mid r \in \mathcal{R} \wedge r' \in \mathcal{R}^*\}$, then we have $\xi_{\overline{\mathcal{M}}}(\langle h_{<t}\rangle) > c\mu(h_{<t})$. □

Next we consider $\overline{\xi}$, the binarised version of $\xi_{\mathcal{M}}$, using Definition 6.1. To show the dominance of $\overline{\xi}$ we have to show that there exists a valid set of priors such that $\xi_{\overline{\mathcal{M}}} = \overline{\xi}$. To this end, we prove the following lemmas about $\mathcal{M}$ and $\overline{\mathcal{M}}$.

Lemma 6.12. *For all $\rho \in \mathcal{M}$, $\forall o, r, h, a.\ r \notin \{0,1\} \Rightarrow \rho(or|ha) = 0$ if and only if $\overline{\rho} = \rho$.*

Proof. ($\Rightarrow$) If $\forall o, r, h, a.\ r \notin \{0,1\} \Rightarrow \rho(or|ha) = 0$, then from the definition of $\overline{\rho}$, we have for reward 1

$$\begin{aligned}
\overline{\rho}(o_t 1|h_{<t}a_t) &= \sum_{r_t \in \mathcal{R}} \rho(o_t r_t|h_{<t}a_t) r_t \\
&= \sum_{r_t \in \{0,1\}} \rho(o_t r_t|h_{<t}a_t) r_t \\
&= \rho(0|h_{<t}a_t o_t) \cdot 0 + \rho(o_t 1|h_{<t}a_t) \cdot 1 \\
&= \rho(o_t 1|h_{<t}a_t),
\end{aligned}$$

and for reward 0

$$\begin{aligned}
\overline{\rho}(o_t 0|h_{<t}a_t) &= \left(\sum_{r_t \in \mathcal{R}} \rho(o_t r_t|h_{<t}a_t) \right) - \overline{\rho}(o_t 1|h_{<t}a_t) \\
&= \left(\sum_{r_t \in \{0,1\}} \rho(o_t r_t|h_{<t}a_t) \right) - \overline{\rho}(o_t 1|h_{<t}a_t) \\
&= \left(\sum_{r_t \in \{0,1\}} \rho(o_t r_t|h_{<t}a_t) \right) - \rho(o_t 1|h_{<t}a_t) \\
&= \rho(o_t 0|h_{<t}a_t).
\end{aligned}$$

Therefore $\overline{\rho} = \rho$.

($\Leftarrow$) If $\overline{\rho} = \rho$, then on all $r \notin \{0,1\}$ we have $\rho(or|ha) = \overline{\rho}(or|ha) = 0$. □

Lemma 6.13. $\overline{\mathcal{M}} = \{\overline{\mu} \mid \mu \in \mathcal{M}\}$.

Proof. We will prove this by showing that $\{\overline{\mu}|\mu \in \mathcal{M}\} \subseteq \overline{\mathcal{M}}$ and $\overline{\mathcal{M}} \subseteq \{\overline{\mu}|\mu \in \mathcal{M}\}$.

For all $\mu \in \mathcal{M}$, we have that $\overline{\mu}$ only produces binary rewards. This means that all $\overline{\mu} \in \overline{\mathcal{M}}$, and therefore $\{\overline{\mu}|\mu \in \mathcal{M}\} \subseteq \overline{\mathcal{M}}$.

We have $\overline{\mathcal{M}} = \{\mu \in \mathcal{M}|\mu \text{ only outputs reward 0 and 1}\} = \{\mu \in \mathcal{M}|\forall o, r, h, a.\ r \notin \{0,1\} \Rightarrow \mu(or|ha) = 0\}$. Therefore, for an arbitrary $\rho \in \overline{\mathcal{M}} = \{\mu \in \mathcal{M}|\forall o, r, h, a.\ r \notin \{0,1\} \Rightarrow \mu(or|ha) = 0\}$ we also have that $\rho \in \mathcal{M}$. Since ρ only outputs rewards of 0 and 1, we also have by Lemma 6.12 that $\overline{\rho} = \rho$, and $\overline{\rho} \in \{\overline{\mu}|\mu \in \mathcal{M}\}$, therefore $\rho \in \{\overline{\mu}|\mu \in \mathcal{M}\}$, hence $\overline{\mathcal{M}} \subseteq \{\overline{\mu}|\mu \in \mathcal{M}\}$. Since we also have that $\{\overline{\mu}|\mu \in \mathcal{M}\} \subseteq \overline{\mathcal{M}}$, it must be the case that $\{\overline{\mu}|\mu \in \mathcal{M}\} = \overline{\mathcal{M}}$. □

Lemma 6.14. *For all sets of lower semicomputable priors $\{w_\mu\}_{\mu \in \mathcal{M}}$, for the Bayes Mixture $\xi_{\mathcal{M}}$, there exists a set of lower semicomputable priors $\{w'_\rho\}_{\rho \in \overline{\mathcal{M}}}$ for the Bayes Mixture $\xi_{\overline{\mathcal{M}}}$ such that for all $h_{1:t} \in \mathcal{H}$ we have*

$$\xi_{\overline{\mathcal{M}}}(\langle h_{1:t} \rangle) = \overline{\xi}_{\mathcal{M}}(\langle h_{1:t} \rangle).$$

Proof. Proof idea: We define a binary environment prior that is the sum of the corresponding non-binary environment priors.

Since $\overline{\mathcal{M}} = \{\overline{\mu} | \mu \in \mathcal{M}\}$, for every element $\rho \in \overline{\mathcal{M}}$ there may be some (possiblly an infinite number of) $\mu \in \mathcal{M}$ such that $\overline{\mu} = \rho$.

If we set the prior $w'_\rho = \sum_{\mu \in \mathcal{M}: \overline{\mu} = \rho} w_\mu$, we will get equality between the mixtures $\xi_{\overline{\mathcal{M}}}(o\langle r \rangle | \langle h_{<t} \rangle a) = \overline{\xi}_{\mathcal{M}}(o\langle r \rangle | \langle h_{<t} \rangle a)$.

To show this equality we will first need to prove that

$$\sum_{\rho \in \overline{\mathcal{M}}} w'_\rho \prod_{k=1}^{t-1} \rho(o_i \langle r_i \rangle | \langle h_{<i} \rangle a_i) = \sum_{\mu \in \mathcal{M}} w_\mu \prod_{k=1}^{t-1} \overline{\mu}(o_i \langle r_i \rangle | \langle h_{<i} \rangle a_i). \tag{6.2}$$

To do this we just expand our definition of w'_ρ and rearrange.

$$\begin{aligned}
\sum_{\rho \in \overline{\mathcal{M}}} w'_\rho \prod_{k=1}^{t-1} \rho(o_i \langle r_i \rangle | \langle h_{<i} \rangle a_i) &= \sum_{\rho \in \overline{\mathcal{M}}} \left(\sum_{\mu \in \mathcal{M}: \overline{\mu} = \rho} w_\mu \right) \prod_{k=1}^{t-1} \rho(o_i \langle r_i \rangle | \langle h_{<i} \rangle a_i) \\
&= \sum_{\rho \in \overline{\mathcal{M}}} \left(\sum_{\mu \in \mathcal{M}: \overline{\mu} = \rho} w_\mu \prod_{k=1}^{t-1} \rho(o_i \langle r_i \rangle | \langle h_{<i} \rangle a_i) \right) \\
&= \sum_{\rho \in \overline{\mathcal{M}}} \left(\sum_{\mu \in \mathcal{M}: \overline{\mu} = \rho} w_\mu \prod_{k=1}^{t-1} \overline{\mu}(o_i \langle r_i \rangle | \langle h_{<i} \rangle a_i) \right) \\
&= \sum_{\mu \in \mathcal{M}} w_\mu \prod_{k=1}^{t-1} \overline{\mu}(o_i \langle r_i \rangle | \langle h_{<i} \rangle a_i).
\end{aligned}$$

Now that we have Equation 6.2 we can prove equality of $\xi_{\overline{\mathcal{M}}}$ and $\overline{\xi}$.

$$\begin{aligned}
\xi_{\overline{\mathcal{M}}}(o\langle r \rangle | \langle h_{<t} \rangle a) &= \frac{\sum_{\rho \in \overline{\mathcal{M}}} w'_\rho \rho(o\langle r \rangle | \langle h_{<t} \rangle a) \prod_{k=1}^{t-1} \rho(o_i \langle r_i \rangle | \langle h_{<i} \rangle a_i)}{\sum_{\nu \in \overline{\mathcal{M}}} w'_\nu \prod_{k=1}^{t-1} \nu(o_i \langle r_i \rangle | \langle h_{<i} \rangle a_i)} \\
&= \frac{\sum_{\mu \in \mathcal{M}} w_\mu \overline{\mu}(o\langle r \rangle | \langle h_{<t} \rangle a) \prod_{k=1}^{t-1} \overline{\mu}(o_i \langle r_i \rangle | \langle h_{<i} \rangle a_i)}{\sum_{\mu \in \mathcal{M}} w_\mu \prod_{k=1}^{t-1} \overline{\mu}(o_i \langle r_i \rangle | \langle h_{<i} \rangle a_i)} \\
&= \overline{\xi}(o\langle r \rangle | \langle h_{<t} \rangle a).
\end{aligned}$$

Therefore for all histories $h_{1:t} \in \mathcal{H}$

$$\xi_{\overline{\mathcal{M}}}(\langle h_{1:t} \rangle) = \prod_{k=1}^{t} \xi_{\overline{\mathcal{M}}}(o_k \langle r_k \rangle | \langle h_{<k} \rangle a_k) = \prod_{k=1}^{t} \overline{\xi}(o_k \langle r_k \rangle | \langle h_{<k} \rangle a_k) = \overline{\xi}(\langle h_{1:t} \rangle).$$

We now need to show that the chosen prior w'_ρ is lower semicomputable.

To compute w'_ρ, we go through the enumeration of $\mu \in \mathcal{M}$ and do a dovetailed proof search on $\overline{\mu} = \rho$, then if we have a proof that $\overline{\mu} = \rho$ we add w_μ to the sum. If there exists a proof such that $\overline{\mu} = \rho$, we will find it in finite steps of the proof search. Since w_μ is lower semicomputable, the sum will also be lower semicomputable. □

Finally, using Lemma 6.14 we can show that binarised reward transformation of the Bayesian mixture $\xi_{\mathcal{M}}$ is, like $\xi_{\overline{\mathcal{M}}}$, a dominant semimeasure in $\mathcal{M}$.

Theorem 6.15. *$\overline{\xi}$ is a dominant semimeasure in $\mathcal{M}$*

Proof. This is a direct consequence of Theorem 6.11 and Lemma 6.14. □

6.6 On Action Binarisation

An effective binarisation of action was proposed by Majeed and Hutter (2021). In this section we build on the results by extending their action binarisation the the Bayesian reinforcement learning case.

To start with, Majeed and Hutter (2021) use a bijective decoder function $D : \{0,1\}^d \to \mathcal{A}$ and an encoder function $C : \mathcal{A} \to \{0,1\}^d$ to decode and encode binary sequences to actions. For the sake of notational simplicity, for this section we will assume $\mathcal{A} = \{0,1\}^d$ since we are not specifically concerned with what the original action space looks like, but splitting it up to be a binary action space over d timesteps.

In this binarised action setup, the binary action history space is defined as follows:

$$\breve{\mathcal{H}} := \bigcup_{t=1}^{\infty} (\mathcal{O} \times \mathcal{R} \times \{0,1\})^{(t-1)} \times \mathcal{O} \times \mathcal{R}\,.$$

Now we can consider a recursive history translation function which takes regular histories and translates them into binary action histories.

Definition 6.16 (Majeed and Hutter (2021))**.** *The history transformation function is expressed with $g : \mathcal{H} \to \breve{\mathcal{H}}$. The map is recursively defined for any history h, action a, next observation o' and next reward r' as*

$$g(hao'r') := g(h)\boldsymbol{a}_1 o_\perp r_\perp \boldsymbol{a}_2 o_\perp r_\perp \ldots \boldsymbol{a}_d o'r' \text{ and } g(e) := e$$

where $\boldsymbol{a} \in \mathcal{A} = \{0,1\}^d$, $o_\perp$ is a "blank" observation of the history, e denotes the "initial" history, and $r_\perp$ is any fixed real-value. We assume $r_\perp = 0 \in \mathcal{R}$.

For shorthand we will use $\overline{\boldsymbol{a}\boldsymbol{o}\boldsymbol{r}_\perp}_{<i}$ to represent $\boldsymbol{a}_1 o r_\perp \ldots \boldsymbol{a}_{i-1} o r_\perp$. Now that we have a method to convert regular histories to binary action histories, we can construct the binary action equivalent of an arbitrary environment μ.

Definition 6.17 (Majeed and Hutter (2021))**.** *For any action $a \in \{0,1\}$, sequentialised history $\tau \in \breve{\mathcal{H}}$, and any partial extension $\overline{\boldsymbol{a}\boldsymbol{o}\boldsymbol{r}_\perp}_{<i}$ for $i < d$, the probability of receiving o' and r' as the next observation and reward is as follows:*

$$\breve{\mu}(o'r'|\tau\overline{\boldsymbol{a}\boldsymbol{o}\boldsymbol{r}_\perp}_{<i}a) := \begin{cases} \mu(o'r'|ha) & \text{if } \tau\overline{\boldsymbol{a}\boldsymbol{o}\boldsymbol{r}_\perp}_{<i}ao'r' = g(hao'r') \\ 1 & \text{if } o'r' = o_\perp r_\perp \\ & \wedge\, g^{-1}(\tau\overline{\boldsymbol{a}\boldsymbol{o}\boldsymbol{r}_\perp}_{<i}ao'r') = \perp \\ 0 & \text{otherwise} \end{cases}$$

where μ is the original environment.

Then to show that $\check{\mu}$ is indeed the binary action equivalent of μ, the following equivalence result was proven.

Theorem 6.18 (Majeed and Hutter (2021))**.** *For any $o' \in \mathcal{O}$, $r' \in \mathcal{R}$, $h \in \mathcal{H}$, and $\boldsymbol{a} \in \mathcal{A}$, the following holds between $\check{\mu}$ and μ*

$$\check{\mu}(o'r'|g(h)\overline{\boldsymbol{a}\boldsymbol{o}\boldsymbol{r}}_{\perp<d}\boldsymbol{a}_d) = \mu(o'r'|h\boldsymbol{a}) \tag{6.3}$$

Extending on this result we have a corollary relating the binarised action environment and original environment, which we will use in a later proof.

Corollary 6.19.

$$\check{\mu}(g(h_{<t})) = \mu(h_{<t}).$$

Proof.

$$\begin{aligned}
\check{\mu}(g(h_{<t})) &\overset{(a)}{=} \check{\mu}(g(h_{<(t-1)})\boldsymbol{a}_1 o r_\perp \boldsymbol{a}_2 o r_\perp \ldots \boldsymbol{a}_d o_{t-1+d} r_{t-1+d}) \\
&= \prod_{i=1}^{t-1}\prod_{k=0}^{d} \check{\mu}(o_{i+k}r_{i+k}|g(h_{<i})\overline{\boldsymbol{a}\boldsymbol{o}\boldsymbol{r}}_{\perp<k}\boldsymbol{a}_k) \\
&\overset{(b)}{=} \prod_{i=1}^{t-1} \check{\mu}(o_i r_i|g(h_{<i})\overline{\boldsymbol{a}\boldsymbol{o}\boldsymbol{r}}_{\perp<d}\boldsymbol{a}_d) \\
&\overset{(c)}{=} \prod_{i=1}^{t-1} \mu(o_i r_i|h_{<i}a_i) \\
&= \mu(h_{<t}).
\end{aligned}$$

(a) comes from the definition of g. (b) comes from Definition 6.17, where it will always be 1 on histories of that form. (c) comes from Theorem 6.18. □

It was additionally shown that in the binarised action setting, a policy which performs well will still perform well in the original setting.

Theorem 6.20 (Majeed and Hutter (2021))**.** *Any $\gamma^{(d-1)/d} \cdot \varepsilon$-optimal policy of the binarised action environment is ε-optimal in the original environment.*

Now considering the Bayesian agent, we are interested in knowing if the learning component of the agent, ξ, will still learn as well in the binary action environment. To this end, let $\check{\mathcal{M}} := \{\mu \in \mathcal{M} \,|\, \mathcal{A} = \{0,1\}\}$ be the class of environments with binary action space. We can show that $\xi_{\check{\mathcal{M}}}$ is indeed a dominant semimeasure over $\mathcal{M}$, that is, it dominates all environments in $\mathcal{M}$.

Theorem 6.21. *$\xi_{\check{\mathcal{M}}}$ is a dominant semimeasure in $\mathcal{M}$.*

Proof. Let $\rho \in \mathcal{M}$. We want to show there exists a constant c such that for all h we have $\xi_{\check{\mathcal{M}}}(g(h)) > c\rho(h)$.

$$\begin{aligned}\xi_{\check{\mathcal{M}}}(g(h)) &= \sum_{\mu \in \check{\mathcal{M}}} w(\mu)\mu(g(h)) \\ &> w(\check{\rho})\check{\rho}(g(h)) \\ &= c\rho(h)\end{aligned}$$

Since $\check{\rho}$ is an environment that only takes binary actions, this completes the proof. □

We can alternatively consider when Definition 6.17 is applied to the original mixture $\xi_{\mathcal{M}}$. In this case we also see that $\check{\xi}$ is a dominant semimeasure over $\mathcal{M}$.

Theorem 6.22. *$\check{\xi}$ is a dominant semimeasure in $\mathcal{M}$.*

Proof. This comes straight from Theorem 6.18 which gives us $\check{\xi}(g(h)) = \xi_{\mathcal{M}}(h)$ □

6.7 Binary Reward (Multiple Bits)

In the previous sections we have discussed how we can turn the reward into a single bit and the effect of this on the environment and the agent. In this section we will discuss various methods for transforming the reward into several bits, but only giving the agent a single bit at a time.

We will start with the overall formulation, and then later go on to various binarising methods and their effects.

Let $\widehat{\mathcal{H}}$ denote the set of histories with only binary rewards, that is

$$\widehat{\mathcal{H}} := \bigcup_{t=0}^{\infty} (\mathcal{A} \times \mathcal{O} \times \{0,1\})^t.$$

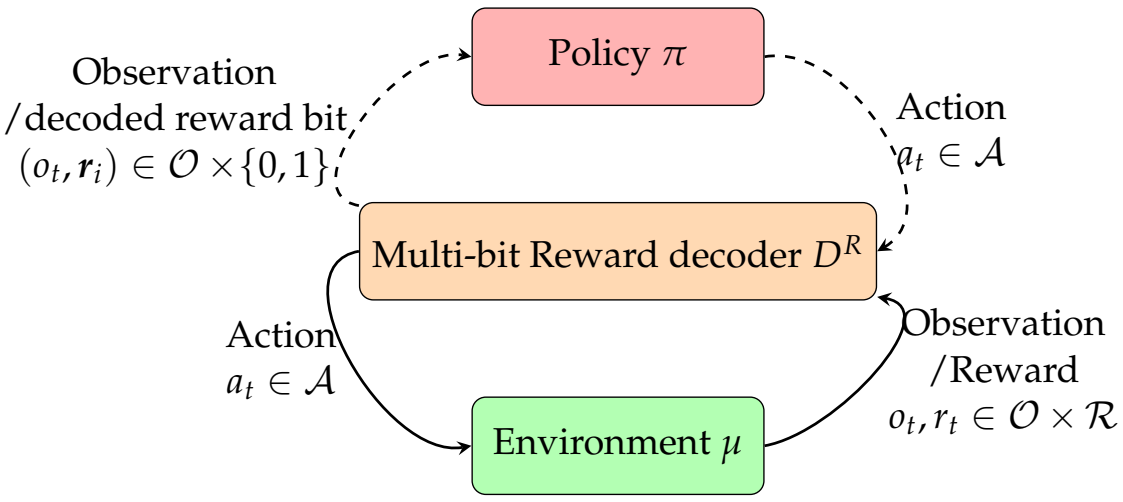

Figure 6.3: Agent-environment loop with multi-bit reward.

We will define an environment transformation $\widehat{\mu}$ in a similar fashion to Definition 6.17, starting with a function which maps multiple-bit binarised histories to original histories. Figure 6.3 shows the setup.

We are going to split up the multiple-bit binary reward environment into two parts: the first part, $\widehat{\mu}_1$, at the point when the true environment outputs the true reward and true observation, and the other part, $\widehat{\mu}_2$, when reward is being decoded. We will use $\boldsymbol{r}$ for the binary form of the true reward from the environment $\mathcal{R}$. We will use d for the length of the binarised rewards; $\boldsymbol{r}$ will always have length d. We will use o for the true observation from the environment, while decoding the reward $\widehat{\mu}_2$ will produce the current observation. We will use the function $\psi : \widehat{\mathcal{H}} \to \mathcal{H}$ to represent translations from the binarised reward histories to the original histories.

$$\psi(h_{1:td}) := \psi(h_{1:(t-1)d})a_{(t-1)d}o_{(t-1)d}C^R(r_{(t-1)d:td})$$
$$\text{and } \psi(\epsilon) = \epsilon$$

and $\psi(h) = \perp$ (undefined) if $\ell(h) \neq td$ for some t, or if $o_{(j-1)d+i} \neq o_{(j-1)d}$ for any i such that $0 \leq i \leq d-1$ and any j such that $1 \leq j \leq t$. The function $C^R : \{0,1\}^d \to \mathcal{R}$ turns the binary form of the rewards to the real-numbered reward in $\mathcal{R}$. This is fine, since we will only be using ψ in cases when the histories have length td for some t.

The inverse of ψ, ψ^{-1}, is not well defined since the choices of actions in the expansion of the history may not match the choices of actions of the agent.

Definition 6.23 (Environment with Binarisation of Rewards into Several Bits)**.** *Given history $h_{1:td}(aor)_{<k}$, if for all $1 \leq j \leq t$ we have $\psi(h_{1:jd}) \neq \perp$, then*

$$\widehat{\mu}(h_{1:td}(aor)_{<k}) := \prod_{j=1}^{t-1} \mu(o_{jd}C^R(r_{jd:(j+1)d})|\psi(h_{1:jd})a_{jd+1})$$
$$\prod_{i=1}^{k-1} \sum_{\boldsymbol{r}\in\mathcal{R}, o\in\mathcal{O}} \mu(o\boldsymbol{r}|\psi(h_{1:td})a_{td+1})[\![r_{td+i} = \boldsymbol{r}_i]\!][\![o_{td+i} = o]\!]$$

else $\widehat{\mu}(h_{1:td}(aor)_{<k}) = 0$.

In this setup we are using a fixed number of maximum bits for the binary encodings of the rewards. This lets us include any possible choices for rewards in $[0,1]$ with the caveat that we may not be able to represent all the rewards exactly.

6.7.1 The problem of $\gamma(t)$

In the multiple-bit binary reward setting, the agent receives these multiple bits over multiple timesteps in the form of single bits. At a given timestep the agent receives a single bit corresponding to the part of the multiple bits representing the reward; the agent also considers the future reward bits it will receive. Specifically, it does so with a discount factor $\gamma(t)$. This

means that the agent does not consider the multiple-bit reward of, for example, 10010 as 0.10010, but instead as $1 + \gamma(t) \cdot 0 + \gamma(t+1) \cdot 0 + \gamma(t+2) \cdot 1 + \gamma(t+3) \cdot 0$.

In binary we could have the multiple bit rewards 100 and 011, and if $\gamma(t) = 1$ then $1 + 1 \cdot 0 + 1 \cdot 0 < 0 + 1 \cdot 1 + 1 \cdot 1$, even though $100 > 011$. One simple solution would be to require $\gamma(t) \geq 2\gamma(t+1)$ for all t. This is a strong assumption, and will mean that the agent does not plan very far ahead. We can be a bit more clever about it.

If we start with some discount function $\gamma(t)$, we can use an updated discount function $\gamma'(t) := \gamma(n) \cdot 2^{-m}$ where $t = d \cdot n + m$ and $m < d$ (d is the number of bits we use). This will give us exact representation if the multiple-bit reward is the binary form of the reward.

An interesting (and unfortunate) consequence is that with γ', we do not necessarily have that $\gamma'(t) \geq \gamma'(t+1)$. However as long as γ is summable, γ' will be summable.

6.7.2 Multiple-bit value function

We can show that if the policy "does not care" about actions it took during the reward decoding then we can get equality in the value functions, since the multiple-bit binary environment automatically does not care about the actions taken during the reward decoding.

Let $\widehat{V}$ be the value function V, with γ replaced by γ'.

Theorem 6.24. *If π is such that $\pi(a|h) = \pi(a|\psi(h))$ for all a, and all histories $h \in \widehat{\mathcal{H}}$ of length td for some t, then we have that for all environments μ and histories $h_{1:td} \in \widehat{\mathcal{H}}$,*

$$\widehat{V}^{\pi,m}_{\widehat{\mu}}(h_{1:td}) = V^{\pi,m}_{\mu}(\psi(h_{1:td})). \tag{6.4}$$

We first need to prove the following lemma. Additionally we have from the definition of ψ that $\psi(h_{1:td}\overline{\boldsymbol{aor}}) = \psi(h_{1:td})\boldsymbol{a}_0 o\boldsymbol{r}$.

Lemma 6.25. *We can write the binarised value function, $\widehat{V}^{\pi,m}_{\widehat{\mu}}$, for d steps with the recursive form of*

$$\widehat{V}^{\pi,m}_{\widehat{\mu}}(h_{1:td}) = \sum_{\boldsymbol{a}\in\mathcal{A}^d} \sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}} \prod_{k=0}^{d-1} \pi(\boldsymbol{a}_k|h_{1:(td+k)})\mu(o\boldsymbol{r}|\psi(h_{1:td})\boldsymbol{a}_0) \left(\gamma(t)\boldsymbol{r} + \widehat{V}^{\pi,m}_{\widehat{\mu}}(h_{1:td}\overline{\boldsymbol{aor}})\right) \tag{6.5}$$

where $\overline{\boldsymbol{aor}} = \boldsymbol{a}_0 o\boldsymbol{r}_0 \ldots \boldsymbol{a}_{d-1} o\boldsymbol{r}_{d-1}$.

Proof.

$$
\begin{aligned}
\widehat{V}^{\pi,m}_{\widehat{\mu}}(h_{1:td}) &= \sum_{a_{td}\in\mathcal{A}} \pi(a_{td}|h_{1:td}) \sum_{o_{td},r_{td}} \widehat{\mu}_1(o_{td}r_{td}|h_{1:td}a_{td},\boldsymbol{r},o)\left(\gamma'(td)r_{td} + \widehat{V}^{\pi,m}_{\widehat{\mu}}(h_{1:td}a_{td}o_{td}r_{td})\right) \\
&= \sum_{a_{td}\in\mathcal{A}} \sum_{o_{td},r_{td}} \cdots \sum_{a_{(t+1)d-1}\in\mathcal{A}} \sum_{o_{(t+1)d-1},r_{(t+1)d-1}} \\
&\prod_{k=0}^{d-1} \pi(a_{td+k}|h_{1:(td+k)}) \prod_{k=0}^{d-1} \widehat{\mu}_2(o_{td+k}r_{td+k}|h_{1:(td+k)}a_{td+k},\boldsymbol{r},o) \\
&\cdot \left(\sum_{i=1}^{d} \gamma'(td+i)r_{td+i} + \widehat{V}^{\pi,m}_{\widehat{\mu}}(h_{<(t+1)d})\right) \\
&= \sum_{a_{td}\in\mathcal{A}} \cdots \sum_{a_{(t+1)d-1}\in\mathcal{A}} \sum_{o_{td},\boldsymbol{r}} \\
&\prod_{k=0}^{d-1} \pi(a_{td+k}|h_{1:(td+k)}) \mu(o_{td}\boldsymbol{r}|\psi(h_{1:td})a_{td}) \left(\gamma(t)\boldsymbol{r} + \widehat{V}^{\pi,m}_{\widehat{\mu}}(h_{1:(t+1)d})\right) \\
&= \sum_{\boldsymbol{a}\in\mathcal{A}^d} \sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}} \prod_{k=0}^{d-1} \pi(\boldsymbol{a}_k|h_{1:(td+k)}) \mu(o\boldsymbol{r}|\psi(h_{1:td})\boldsymbol{a}_0) \left(\gamma(t)\boldsymbol{r} + \widehat{V}^{\pi,m}_{\widehat{\mu}}(h_{1:td}\overline{\boldsymbol{a}o\boldsymbol{r}})\right).
\end{aligned}
$$

□

Proof of Theorem 6.24.

Proof. Final case, when $t = m$

$$
\begin{aligned}
\widehat{V}^{\pi,m}_{\widehat{\mu}}(h_{1:md}) &= \sum_{\boldsymbol{a}\in\mathcal{A}^d} \sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}} \prod_{k=0}^{d-1} \pi(\boldsymbol{a}_k|h_{1:(md+k)}) \mu(o\boldsymbol{r}|\psi(h_{1:md})\boldsymbol{a}_0)\,(\gamma(m)\boldsymbol{r}) \\
&= \sum_{\boldsymbol{a}_0\in\mathcal{A}} \sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}} \pi(\boldsymbol{a}_0|h_{1:td}) \mu(o\boldsymbol{r}|\psi(h_{1:td})\boldsymbol{a}_0)\,(\gamma(m)\boldsymbol{r}) \\
&= \sum_{\boldsymbol{a}_0\in\mathcal{A}} \pi(\boldsymbol{a}_0|h_{1:td}) \sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}} \mu(o\boldsymbol{r}|\psi(h_{1:td})\boldsymbol{a}_0)\,(\gamma(m)\boldsymbol{r}) \\
&= \sum_{\boldsymbol{a}_0\in\mathcal{A}} \pi(\boldsymbol{a}_0|\psi(h_{1:td})) \sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}} \mu(o\boldsymbol{r}|\psi(h_{1:td})\boldsymbol{a}_0)\,(\gamma(m)\boldsymbol{r}) \\
&= V^{\pi,m}_{\mu}(\psi(h_{1:md})).
\end{aligned}
$$

Then for $t < m$ we have we can work backwards from m,

$$
\begin{aligned}
\widehat{V}^{\pi,m}_{\widehat{\mu}}(h_{1:td}) &= \sum_{\boldsymbol{a}\in\mathcal{A}^d}\sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}}\prod_{k=0}^{d-1}\pi(\boldsymbol{a}_k|h_{1:(td+k)})\mu(o\boldsymbol{r}|\psi(h_{1:td})\boldsymbol{a}_0)\left(\gamma(t)\boldsymbol{r}+\widehat{V}^{\pi,m}_{\widehat{\mu}}(h_{1:td}\overline{\boldsymbol{a}o\boldsymbol{r}})\right)\\
&= \sum_{\boldsymbol{a}\in\mathcal{A}^d}\sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}}\prod_{k=0}^{d-1}\pi(\boldsymbol{a}_k|h_{1:(td+k)})\mu(o\boldsymbol{r}|\psi(h_{1:td})\boldsymbol{a}_0)\left(\gamma(t)\boldsymbol{r}+V^{\pi,m}_{\mu}(\psi(h_{1:td}\overline{\boldsymbol{a}o\boldsymbol{r}}))\right)\\
&= \sum_{\boldsymbol{a}\in\mathcal{A}^d}\sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}}\prod_{k=0}^{d-1}\pi(\boldsymbol{a}_k|h_{1:(td+k)})\mu(o\boldsymbol{r}|\psi(h_{1:td})\boldsymbol{a}_0)\left(\gamma(t)\boldsymbol{r}+V^{\pi,m}_{\mu}(\psi(h_{1:td})\boldsymbol{a}_0o\boldsymbol{r})\right)\\
&= \sum_{\boldsymbol{a}_0\in\mathcal{A}}\sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}}\pi(\boldsymbol{a}_0|h_{1:td})\mu(o\boldsymbol{r}|\psi(h_{1:td})\boldsymbol{a}_0)\left(\gamma(t)\boldsymbol{r}+V^{\pi,m}_{\mu}(\psi(h_{1:td})\boldsymbol{a}_0o\boldsymbol{r})\right)\\
&= \sum_{\boldsymbol{a}_0\in\mathcal{A}}\pi(\boldsymbol{a}_0|h_{1:td})\sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}}\mu(o\boldsymbol{r}|\psi(h_{1:td})\boldsymbol{a}_0)\left(\gamma(t)\boldsymbol{r}+V^{\pi,m}_{\mu}(\psi(h_{1:td})\boldsymbol{a}_0o\boldsymbol{r})\right)\\
&= \sum_{\boldsymbol{a}_0\in\mathcal{A}}\pi(\boldsymbol{a}_0|\psi(h_{1:td}))\sum_{o\in\mathcal{O},\boldsymbol{r}\in\mathcal{R}}\mu(o\boldsymbol{r}|\psi(h_{1:td})\boldsymbol{a}_0)\left(\gamma(t)\boldsymbol{r}+V^{\pi,m}_{\mu}(\psi(h_{1:td})\boldsymbol{a}_0o\boldsymbol{r})\right)\\
&= V^{\pi,m}_{\mu}(\psi(h_{1:td})).
\end{aligned}
$$

□

It could be argued that the assumption that $\pi(a|h) = \pi(a|\psi(h))$ for all a, and all histories $h \in \widehat{\mathcal{H}}$ of length td for some t, is quite strong. However, we found this assumption difficult to relax; essentially one requires some structural assumption on the policy to state anything meaningful.

6.8 Discussion and Future Work

We started with original reward set being a subset of $[0,1]$. This is not a restriction, as if it is the case that the original reward set is $\mathcal{R} = [-a,b]$ then since the optimal policy is invariant under positive scaling to $\mathcal{R} = [0,1]$ there is no difference. If there exists some absorbing state or semi-measure, then reduction to the (binary) reward set $\mathcal{R} = \{-a,b\}$ is still possible, or, reduction to $\mathcal{R} = \{0,1\}$ assuming a special reward is given at the absorbing state or semi-measure "death" state.

Throughout this chapter we have presented results for the value function, however, all results would still hold if we were to consider the action value (Q-value) function instead.

One possible extension of this work would be to try to achieve a result similar to Theorem 6.20 for single-bit or multiple-bit binary rewards. Specifically we may be able to show that an agent which is close to optimal in the binarised case is also close to optimal in the original environment, as the negative result given by Theorem 6.4 only precludes the possibility of exact representation.

It would be useful to, on top of the binary action and reward space, have a binary observation space in one complete framework. This can be done easily for the observation space, however,

the combination of observation and reward space may lead to some problems similar to that of the reward space binarisation.

6.9 Conclusion

In this work we have analysed a number of natural reward binarisation schemes. While in general we show that exact representation of the value function is impossible, we found that reward binarisation can be applied in many important special cases without a loss of representation power. Additionally, we considered the interaction of binarisation with Bayesian reinforcement learning, providing new technical results that justify the Bayesian approach in the presence of specific types of binarisation.

Chapter 7

A Strongly Asymptotically Optimal Agent in General Environments

This chapter has been published as

M. K. Cohen, E. Catt et al. (2019). 'A strongly asymptotically optimal agent in general environments'. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 2179–2186

My main contribution to this paper was with Section 7.5. I have omitted the proofs of the theoretical results as they were not a main component of my contribution. These omitted proofs can be found in the publication.

Abstract

Reinforcement Learning agents are expected to eventually perform well. Typically, this takes the form of a guarantee about the asymptotic behavior of an algorithm given some assumptions about the environment. We present an algorithm for a policy whose value approaches the optimal value with probability 1 in all computable probabilistic environments, provided the agent has a bounded horizon. This is known as strong asymptotic optimality, and it was previously unknown whether it was possible for a policy to be strongly asymptotically optimal in the class of all computable probabilistic environments. Our agent, Inquisitive Reinforcement Learner (Inq), is more likely to explore the more it expects an exploratory action to reduce its uncertainty about which environment it is in, hence the term inquisitive. Exploring inquisitively is a strategy that can be applied generally; for more manageable environment classes, inquisitiveness is tractable. We conducted experiments in "grid-worlds" to compare the Inquisitive Reinforcement Learner to other weakly asymptotically optimal agents.

7.1 Introduction

> *"Efforts to solve [an instance of the exploration-exploitation problem] so sapped the energies and minds of Allied analysts that the suggestion was made that the problem be dropped over Germany, as the ultimate instrument of intellectual sabotage."* –Peter Whittle (Whittle, 1979)

The Allied analysts were considering the simplest possible problem in which there is a trade-off to be made between exploiting, taking the apparently best option, and exploring, choosing a different option to learn more. We tackle what we consider the most difficult instance of the exploration-exploitation trade-off problem: when the environment could be any computable probability distribution, not just a multi-armed bandit, how can one achieve optimal performance in the limit?

Our work is within the Reinforcement Learning (RL) paradigm: an agent selects an action, and the environment responds with an observation and a reward. The interaction may end, or it may continue forever. Each interaction cycle is called a timestep. The agent has a discount function that weights its relative concern for the reward it achieves at various future timesteps. The agent's job is to select actions that maximize the total expected discounted reward it achieves in its lifetime. The "value" of an agent's policy at a certain point in time is the expected total discounted reward it achieves after that time if it follows that policy. One formal specification of the exploration-exploitation problem is: what policy can an agent follow so that the policy's value approaches the value of the optimal informed policy with probability 1, even when the agent doesn't start out knowing the true dynamics of its environment?

Most work in RL makes strong assumptions about the environment—that the environment is Markov, for instance. Impressive recent development in the field of reinforcement learning often makes use of the Markov assumption, including Deep Q Networks (Mnih, Kavukcuoglu et al., 2015), A3C (Mnih, Badia et al., 2016), Rainbow (Hessel et al., 2018), and AlphaZero (Silver, Hubert et al., 2018). Another example of making strong assumptions in RL comes from some model-based algorithms that implicitly assume that the environment is representable by, for example, a fixed-size neural network, or whatever construct is used to model the environment. We do not make any such assumptions.

Many recent developments in RL are largely about tractably learning to exploit; how to explore intelligently is a separate problem. We address the latter problem. Our approach, inquisitiveness, is based on Orseau, Lattimore et al. (2013) Knowledge Seeking Agent for Stochastic Environments, which selects the actions that best inform the agent about what environment it is in. Our Inquisitive Reinforcement Learner (Inq) explores like a knowledge seeking agent, and is more likely to explore when there is apparently (according to its current beliefs) more to be learned. Sometimes exploring well requires "expeditions," or many consecutive exploratory actions. Inq entertains expeditions of all lengths, although it follows the longer ones less often, and it doesn't resolutely commit in advance to seeing the expedition through.

This is a very human approach to information acquisition. When we spot an opportunity to learn something about our natural environment, we feel inquisitive. We get distracted. We are inclined to check it out, even if we don't see directly in advance how this information might help us better achieve our goals. Moreover, if we can tell that the opportunity to learn something requires a longer term project, we may find ourselves less inquisitive.

For the class of computable environments (stochastic environments that follow a computable probability distribution), it was previously unknown whether any policy could achieve strong asymptotic optimality (convergence of the value to optimality with probability 1). Lattimore and Hutter (2011) showed that no deterministic policy could achieve this. The key advantage that stochastic policies have is that they can let the exploration probability go to 0 while still exploring infinitely often. (For example, an agent that explores with probability $1/t$ at time t still explores infinitely often).

There is a weaker notion of optimality–"weak asymptotic optimality"–for which positive results already exist; this condition requires that the average value over the agent's lifetime approach optimality. Lattimore and Hutter (2011) identified a weakly asymptotically optimal agent for *deterministic* computable environments; the agent maintains a list of environments consistent with its observations, exploiting as if it is in the first such one, and exploring in bursts. A recent algorithm for a Thompson Sampling Bayesian agent was shown, with an elegant proof, to be weakly asymptotically optimal in all computable environments, but not strongly asymptotically optimal (Leike, Lattimore et al., 2016). Our algorithm is inspired from Cohen et al.'s algorithm for an extremely myopic agent "Boxed Myopic Artificial Intelligence"; a few lemmas in our proof of strong asymptotic optimality were first shown in that chapter, but we repeat them here for completeness.

Most work in RL regards (Partially Observable) Markov Decision Processes (PO)MDPs. However, environments that enter completely novel states infinitely often render (PO)MDP algorithms helpless. For example, an RL agent acting as a chatbot, optimizing a function, or proving mathematical theorems would struggle to model the environment as an MDP, and would likely require an exploration mechanism like ours. In the chatbot case, for instance, as a conversation with a person progresses, the person never returns to the same state.

If we formally compare Inq to existing algorithms in MDPs, we find that many achieve asymptotic optimality. Epsilon-greedy, upper confidence bound, and Thompson sampling exploration strategies suffice in MDPs. Our primary motivation is for the sorts of environments described above. To discriminate between exploratory approaches in *ergodic* MDPs, one can formally bound regret, and we would like to do this for Inq in the future.

For comparison, some algorithms which use the MDP formalism also consider information-theoretic approaches to exploration, such as VIME (Houthooft et al., 2016), the agent in (Still, 2009), and TEXPLORE-VANIR (Hester and Stone, 2012).

In Section 7.2, we formally describe the RL setup and present notation. In Section 7.3, we present the algorithm for Inq. In Section 7.4, we provide our main result: that Inq is strongly asymptotically optimal. In Section 7.5, we present experimental results comparing Inq to

weakly asymptotically optimal agents. Finally, we discuss the relevance of this exploration regime to tractable algorithms.

7.2 Notation

In this chapter we will use the notation given in chapter 6, with some additions. The set of all possible t length interactions is given by $\mathcal{H}_t := (\mathcal{A} \times \mathcal{O} \times \mathcal{R})^t$. The set of all possible interactions of length less than m is given by $\mathcal{H}_{<m} := \cup_{t=0}^{m-1} \mathcal{H}^t$. A policy and an environment induce a probability measure over $\mathcal{H}_\infty := (\mathcal{A} \times \mathcal{O} \times \mathcal{R})^\infty$, the set of all possible infinite histories: for $h \in \mathcal{H}$, $\mathrm{P}_\nu^\pi(h)$ denotes the probability that an infinite history begins with h when actions are sampled from the policy π, and observations and rewards are sampled from the environment ν. Formally, we define this inductively: $\mathrm{P}_\nu^\pi(\epsilon) \mapsto 1$, where ϵ is the empty history, and for $h \in \mathcal{H}$, $a \in \mathcal{A}$, $o \in \mathcal{O}$, $r \in \mathcal{R}$, we define $\mathrm{P}_\nu^\pi(haor) \mapsto \mathrm{P}_\nu^\pi(h)\pi(a|h)\nu(or|ha)$. In an infinite history $h_{1:\infty} \in \mathcal{H}_\infty$, a_t, o_t, and r_t refer to the tth action, observation and reward, and h_t refers to the tth timestep: $a_t o_t r_t$. $h_{<t}$ refers to the first $t-1$ timesteps, and $h_{t:k}$ refers to the string of timesteps t through k (inclusive). Strings of actions, observations, and rewards are notated similarly.

A Bayesian agent deems a class of environments a priori feasible. Its "beliefs" take the form of a probability distribution over which environment is the true one. We call this the agent's belief distribution. In our formulation, Inq considers any computable environment feasible, and starts with a prior belief distribution based on the environments' Kolmogorov complexities: that is, the length of the shortest program that computes the environment on some reference machine. However, all our results hold as long as the true environment is contained in the class of environments that are considered feasible, and as long as the prior belief distribution assigns nonzero probability to each environment in the class. We take $\mathcal{M}$ to be the class of all computable environments, and $w(\nu) := 2^{-K(\nu)(1+\varepsilon)}/\mathcal{N}$ to be the prior probability of the environment ν, where K is the Kolmogorov complexity, $\varepsilon > 0$, and $\mathcal{N}$ is a normalization constant. ($\varepsilon > 0$ ensures the prior has finite entropy, which facilitates analysis.) A smaller class with a different prior probability could easily be substituted for $\mathcal{M}$ and $w(\nu)$.

We use ξ to denote the agent's beliefs about future observations. Together with a policy π it defines a Bayesian mixture measure: $\mathrm{P}_\xi^\pi(\cdot) := \sum_{\nu \in \mathcal{M}} w(\nu)\,\mathrm{P}_\nu^\pi(\cdot)$. The posterior belief distribution of the agent after observing a history $h \in \mathcal{H}$ is $w(\nu|h) := w(\nu)\,\mathrm{P}_\nu^{\pi'}(h)\,/\,\mathrm{P}_\xi^{\pi'}(h)$. This definition is independent of the choice of π' as long as $\mathrm{P}_\xi^{\pi'}(h) > 0$; we can fix a reference policy π' just for this definition if we like. We sometimes also refer to the conditional distribution $\xi(or|ha) := \sum_{\nu \in \mathcal{M}} w(\nu|h)\nu(or|ha)$.

The agent's discount at a timestep is denoted γ_t. To normalize the agent's policy's value to $[0,1]$, we introduce $\Gamma_t := \sum_{k=t}^\infty \gamma_k$. (Normalization makes value convergence nontrivial). We consider an agent with a bounded horizon: $\forall \varepsilon > 0\ \exists m\ \forall t : \Gamma_{t+m}/\Gamma_t \leq \varepsilon$. Intuitively, this means that the agent does not become more and more farsighted over time. Note this does not require a finite horizon. A classic discount function giving a bounded horizon is a geometric one: for $0 \leq \gamma < 1$, $\gamma_t = \gamma^t$. Recall the value of a policy π in an environment ν,

given a history $h_{<t} \in \mathcal{H}_{t-1}$, is

$$V_\nu^\pi(h_{<t}) := \frac{1}{\Gamma_t}\mathbb{E}_\nu^\pi\left[\sum_{k=t}^{\infty}\gamma_k r_k \;\middle|\; h_{<t}\right] \tag{7.1}$$

Here, the expectation is with respect to the probability measure P_ν^π. Reinforcement Learning is the attempt to find a policy that makes this value high, without access to ν.

7.3 Inquisitive Reinforcement Learner

We first describe how Inq exploits, then how it explores. It exploits by maximizing the discounted sum of its reward in expectation over its current beliefs, and it explores by following maximally informative "exploratory expeditions" of various lengths.

An optimal policy with respect to an environment ν is a policy that maximizes the value.

$$\pi_\nu^*(\cdot) := \operatorname*{argmax}_{\pi\in\Pi} V_\nu^\pi(\cdot) \tag{7.2}$$

where $\Pi = \mathcal{H} \to \Delta\mathcal{A}$ is the space of all policies. An optimal deterministic policy always exists (Lattimore and Hutter, 2014b). When exploiting, Inq simply maximizes the value according to its belief distribution ξ. Since this policy is deterministic, we write $a^*(h_{<t})$ to mean the unique action at time t for which $\pi_\xi^*(a|h_{<t}) = 1$. That is the exploitative action.

The most interesting feature of Inq is how it gets distracted by the opportunity to explore. Inq explores to learn. An agent has learned from an observation if its belief distribution w changes significantly after making that observation. If the belief distribution has hardly changed, then the observation was not very informative. The typical information-theoretic measure for how well a distribution Q approximates a distribution P is the KL-divergence, $\mathrm{KL}(P||Q)$. Thus, a principled way to quantify the information that an agent gains in a timestep is the KL-divergence from the belief distribution at time $t+1$ to the belief distribution at time t. This is the rationale behind the construction of Orseau, Lattimore et al. (2013) Knowledge Seeking Agent, which maximizes this expected information gain.

Letting $h_{<t} \in \mathcal{H}_{t-1}$ and $h' \in \mathcal{H}$, the information gain at time t is defined:

$$\mathrm{IG}(h'|h_{<t}) := \sum_{\nu\in\mathcal{M}} w(\nu|h_{<t}h')\log\frac{w(\nu|h_{<t}h')}{w(\nu|h_{<t})} \tag{7.3}$$

Recall that $w(\nu|h)$ is the posterior probability assigned to ν after observing h.

An m-step expedition, denoted π_m, represents all contingencies for how an agent will act for the next m timesteps. It is a deterministic policy that takes history-fragments of length less than m and returns an action. Let $\Pi_{<m} := \mathcal{H}_{<m} \to \mathcal{A}$ be the set of all m-step expeditions. $\mathrm{P}_\xi^{\pi_m}(h_{<t+k}|h_{<t})$ is a conditional distribution defined for $0 \le k \le m$, which represents the conditional probability of observing $h_{<t+k}$ if the expedition π_m is followed starting at time t, after observing $h_{<t}$. Now we can consider the information-gain value of an m-step expedition.

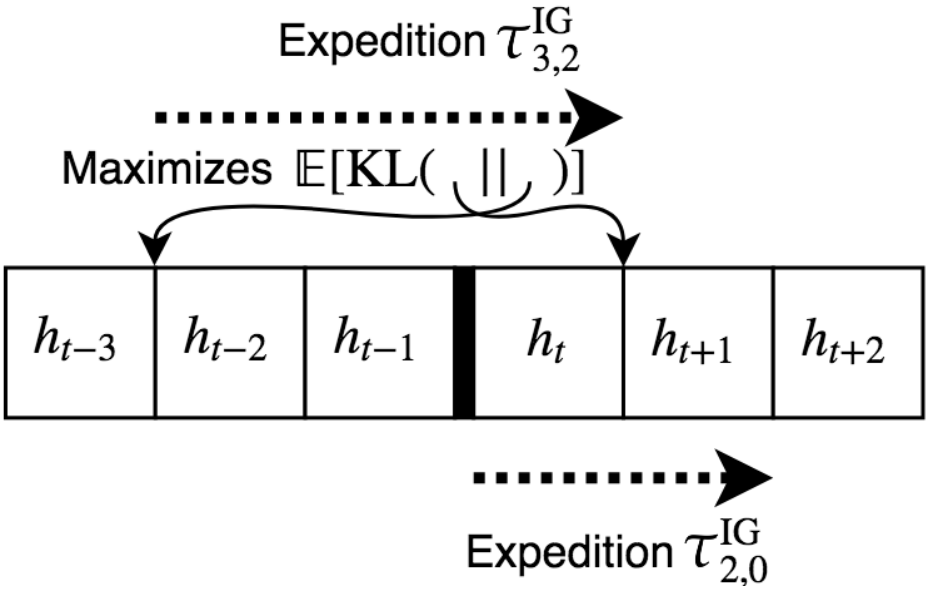

Figure 7.1: Example expeditions. Expeditions maximize the expected KL-divergence from the posterior at the end to the posterior at the beginning.

It is the expected information gain upon following that expedition:

$$V^{\text{IG}}(\pi_m, h_{<t}) := \sum_{h_{t:t+m-1} \in \mathcal{H}_m} \text{P}_\xi^{\pi_m}(h_{<t+m}|h_{<t})\, \text{IG}(h_{t:t+m-1}|h_{<t}) \tag{7.4}$$

At a time t, one might consider many expeditions: the one-step expedition which maximizes expected information gain, the two-step expedition doing the same, etc. Or one might consider carrying on with an expedition that began three timesteps ago.

Definition 7.1. *At time t, the m-k expedition is the m-step expedition beginning at time $t-k$ which maximized the expected information gain from that point.*[1]

$$\tau^{\text{IG}}_{m,k}(h_{<t}) := \operatorname*{argmax}_{\pi_m \in \Pi_{<m}} V^{\text{IG}}(\pi_m, h_{<t-k}) \tag{7.5}$$

Example expeditions are diagrammed in Figure 7.1.

Expeditions are functions which return an action given what has been seen so far on the expedition. The m-k exploratory action is the action to take at time t according to the m-k expedition:

$$a^{\text{IG}}_{m,k}(h_{<t}) := \tau^{\text{IG}}_{m,k}(h_{<t})(h_{t-k:t-1}) \tag{7.6}$$

Naturally, this is only defined for $k < m, t$, since the expedition function can't accept a history fragment of length $\geq m$, and $t-k$ must be positive. Note also that if $k = 0$, $h_{t-k:t-1}$ evaluates to the empty string, ϵ.

[1] Ties in the argmax are broken arbitrarily.

The reason Inq doesn't ignore expeditions that started in the past is that Inq must have some chance of actually executing the whole expedition (for every expedition). If the probability of completing an expedition is 0, one cannot use it for a bound on Inq's belief-accuracy.

Definition 7.2. *Let $\rho(h_{<t}, m, k)$ be the probability of taking the m-k exploratory action after observing a history $h_{<t}$.*

$$\rho(h_{<t}, m, k) := \min\left\{\frac{1}{m^2(m+1)}, \eta V^{\text{IG}}(\tau^{\text{IG}}_{m,k}(h_{<t}), h_{<t-k})\right\}$$

where η is an exploration constant.

Note in the definition of $\rho(h_{<t}, m, k)$ that the probability of following an expedition goes to 0 if the expected information gain from that expedition goes to 0. The first term in the min ensures the probabilities will not sum to more than 1. The total probability of exploration is defined:

$$\beta(h_{<t}) := \sum_{m\in\mathbb{N}}\sum_{k<m,t}\rho(h_{<t}, m, k) \leq \sum_{m\in\mathbb{N}}\sum_{k<m,t}\frac{1}{m^2(m+1)} \leq \sum_{m\in\mathbb{N}}\sum_{k<m}\frac{1}{m^2(m+1)} = 1$$

The feature that makes Inq inquisitive is that $\rho(h_{<t}, m, k)$ is proportional to the expected information gain from the *m-k* expedition, $V^{\text{IG}}(\tau^{\text{IG}}_{m,k}(h_{<t}), h_{<t-k})$. Note that completing an *m*-step expedition requires randomly deciding to explore in that way on *m* separate occasions. While this may seem inefficient, if the agent always got boxed into long expeditions, the value of its policy would plummet infinitely often.

Finally, Inq's policy $\pi^\dagger$, defined in Algorithm 2, takes the *m-k* exploratory action with probability $\rho(\cdot, m, k)$, and takes the exploitative action otherwise.[2]

Algorithm 2 Inquisitive Reinforcement Learner's Policy $\pi^\dagger$

while True **do**
  calculate $\rho(h_{<t}, m, k)$ for all m and for all $k < \min\{m, t\}$
  take action $a^{\text{IG}}_{m,k}(h_{<t})$ with probability $\rho(h_{<t}, m, k)$
  take action $a^*(h_{<t})$ with probability $1 - \beta(h_{<t})$

7.4 Strong Asymptotic Optimality

Here we present our central result: that the value of $\pi^\dagger$ approaches the optimal value. We present the theorem, motivate the result, and proceed to the proof.

Before presenting the theorem, we clarify an assumption, and define the optimal value. We call the true environment μ, and we assume that $\mu \in \mathcal{M}$. For $\mathcal{M}$ the class of computable environments, this is a very unassuming assumption. The optimal value is simply the value

[2]This algorithm is written in a simplified way that does not halt, but if a real number in $[0, 1]$ is sampled first, the actions can be assigned to disjoint intervals successively until the sampled real number lands in one of them.

of the optimal policy with respect to the true environment:

$$V_\mu^*(h_{<t}) := \sup_{\pi \in \Pi} V_\mu^\pi(h_{<t}) = V_\mu^{\pi_\mu^*}(h_{<t}) \tag{7.7}$$

Recall also that we have assumed the agent has a bounded horizon in the sense that $\forall \varepsilon \; \exists m \; \forall t : \Gamma_{t+m}/\Gamma_t \leq \varepsilon$. The Strong Asymptotic Optimality theorem is that under these conditions, the value of Inq's policy approaches the optimal value with probability 1, when actions are sampled from Inq's policy and observations and rewards are sampled from the true environment μ.

Theorem 7.3 (Strong Asymptotic Optimality)**.** *As $t \to \infty$,*

$$V_\mu^*(h_{<t}) - V_\mu^{\pi^\dagger}(h_{<t}) \to 0 \;\; \textit{with} \;\; \mathrm{P}_\mu^{\pi^\dagger}\textit{-prob. 1}$$

where $\mu \in \mathcal{M}$ is the true environment.

For a Bayesian agent, uncertainty about on-policy observations goes to 0. Since "on-policy" for Inq includes, with some probability, all maximally informative expeditions, Inq eventually has little uncertainty about the result of any course of action, and can therefore successfully select the optimal course. For any fixed horizon, Inq's mixture measure ξ approaches the true environment μ.

Strong Asymptotic Optimality is not a guarantee of efficacy; consider an agent that "commits suicide" on the first timestep, and thereafter receives a reward of 0 no matter what it does. This agent is asymptotically optimal, but not very useful. In general, when considering many environments with many different "traps," bounded regret is impossible to guarantee (Hutter, 2005), but one can still demand from a reinforcement learner that it make the best of whatever situation it finds itself in by correctly identifying (in the limit) the optimal policy.

We suspect that strong asymptotic optimality would not hold if Inq had an unbounded horizon, since its horizon of concern may grow faster than it can learn about progressively more long-term dynamics of the environment. Going more into the technical details, let Δ_{kt} be, roughly "at time t, how much does ξ differ from μ regarding predictions about the next k timesteps?" A lemma in our proof is that $\forall k \;\; \lim_{t\to\infty} \Delta_{kt} = 0$, but this does not imply, for example, that $\lim_{z\to\infty} \Delta_{zz} = 0$. If the horizon which is necessary to predict is growing over time, Inq might not be strongly asymptotically optimal.

Indeed, we tenuously suspect that it is impossible for an agent with an unbounded time horizon to be strongly asymptotically optimal in the class of all computable environments. If that is true, then the assumptions that our result relies on (namely that the true environment is computable, and the agent has a bounded horizon) are the bare minimum for strong asymptotic optimality to be possible.

Inq is not computable; in fact, no computable policy can be strongly asymptotically optimal in the class of all computable environments (Lattimore and Hutter (2011) show this for deterministic policies, but a simple modification extends this to stochastic policies). For many

smaller environment classes, however, Inq would be computable, for example if $\mathcal{M}$ is finite, and perhaps for decidable $\mathcal{M}$ in general. The central result, that inquisitiveness is an effective exploration strategy, applies to any Bayesian agent.

7.5 Experimental Results

We compared Inq with other known weakly asymptotically optimal agents, Thompson sampling and BayesExp (Lattimore and Hutter, 2014a), in the grid-world environment using AIXIjs (Aslanides, 2017) which has previously been used to compare asymptotically optimal agents (Aslanides et al., 2017). We tested in 10×10 grid-worlds, and 20×20 grid-worlds, both with a single dispenser with probability of dispensing reward 0.75; that is, if the agent enters that cell, the probability of a reward of 1 is 0.75. Following the conventions of Aslanides et al. (2017) we averaged over 50 simulations, used discount factor $\gamma = 0.99$, 600 MCTS samples, and planning horizon of 6. The planning horizon restricts m, and the number of MCTS samples is an input to ρUCT (Silver and Veness, 2010), which we use instead of expectimax. The algorithm for the approximate version of Inq is in Section 7.5.1. The code used for this experiment is available online at `https://github.com/ejcatt/aixijs`, and this version of Inq can be run in the browser at `https://catt.id/aixijs/demo.html#inq`. We found that using small values for η, specifically $\eta \leq 1$ worked well. For our experiments we chose $\eta = 1$.

In the 10×10 grid-worlds Inq performed comparably to both BayesExp and Thompson sampling. However in the 20×20 grid-worlds Inq performed comparably to BayesExp, and outperformed Thompson sampling. This is likely because when the Thomspon Sampling Agent samples an environment with a reward dispenser that is inaccessible within its planning horizon, the agent acts randomly rather than seeking new cells. This is in contrast to Inq and BayesExp which always have an incentive to explore the frontier of cells that have not been visited. This is especially relevant in the larger grid where the Thomspon sampling agent is more likely to act as if the dispenser is deep in uncharted territory, rather than nearby. In a grid-world, good exploration is just about visiting new states, which both Inq and BayesExp successfully seek.

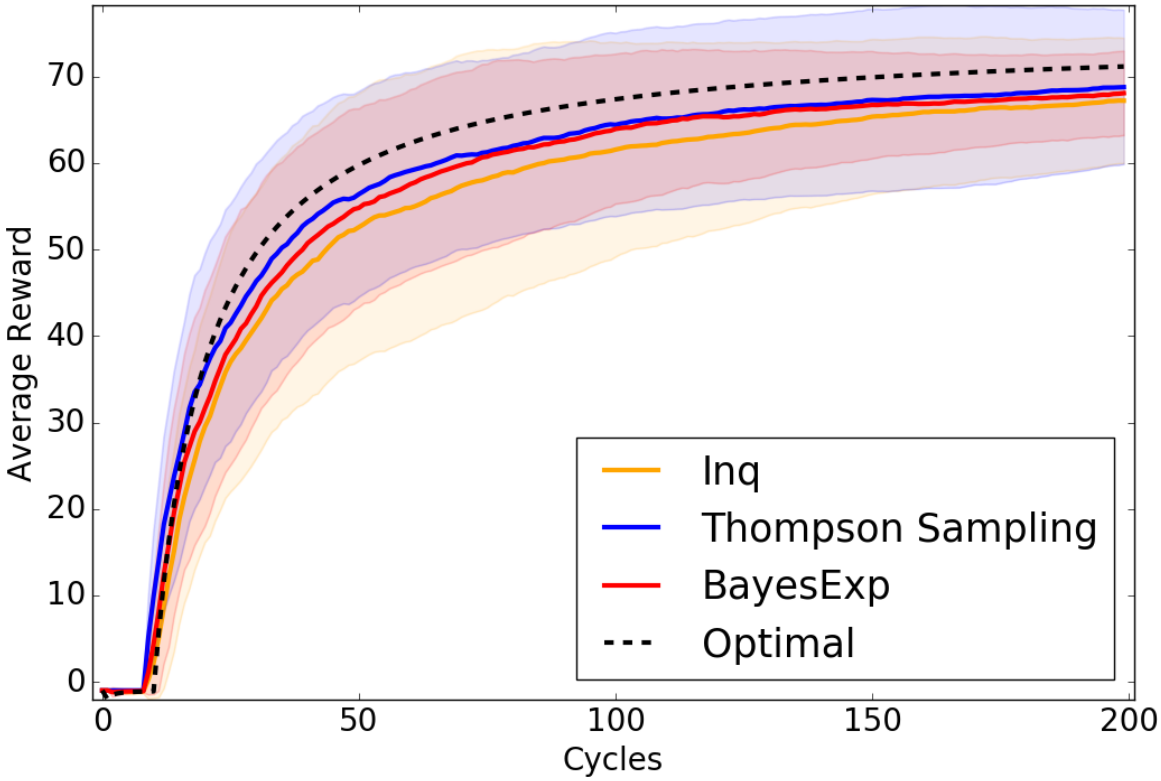

Figure 7.2: 10×10 Grid-worlds

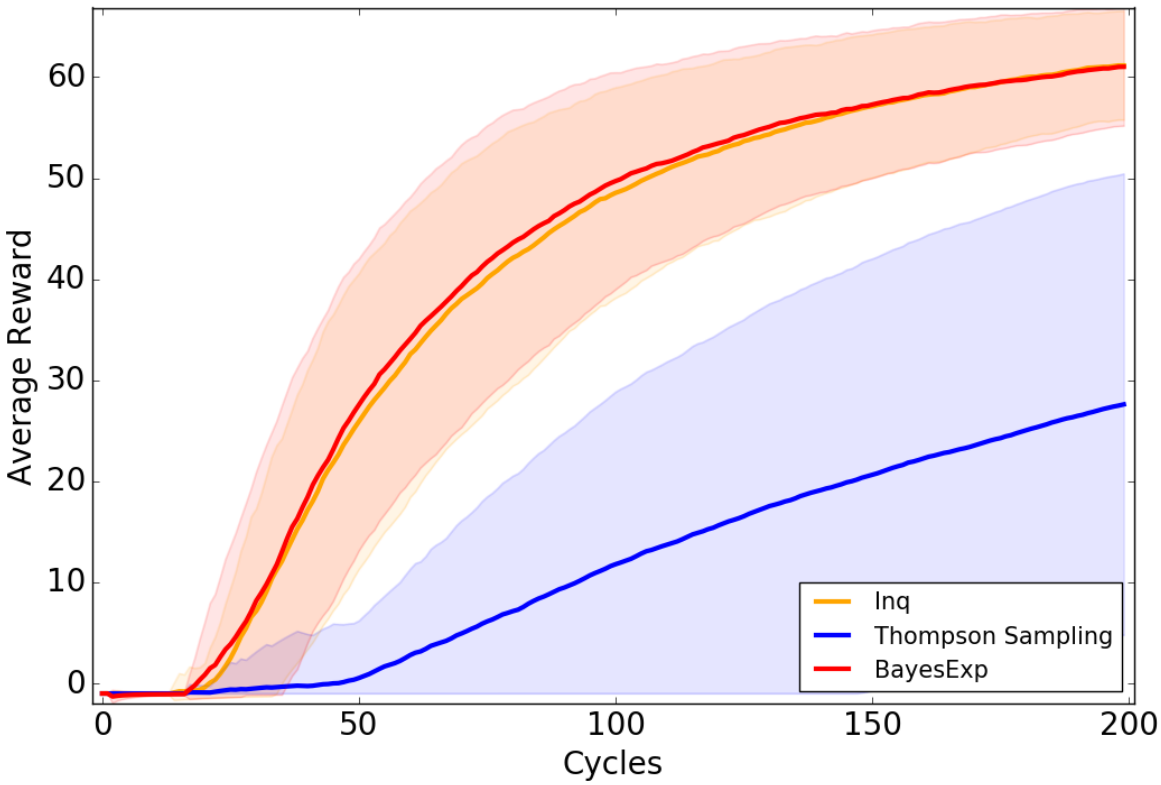

Figure 7.3: 20×20 Grid-worlds

7.5.1 Approximation of Inq

Following Aslanides (2017), our approximation of Inq calls ρUCT (Silver and Veness, 2010) as a subroutine in place of expectimax.

Algorithm 3 Approximation of Inquisitive Reinforcement Learner's Policy

Require: MCTS Samples, horizon, γ
Initialize: uniform prior over model class

while True **do**
  for all $m \leq$ horizon and $k < \min\{m, t\}$ **do**
    using information gain as reward, $a^{\text{IG}}_{m,k} \sim \rho\text{UCT}(h_{<t-k}, \text{MCTS samples}, m, \gamma)$
    $\rho(m,k) = \min\{\text{information-gain-value of } a^{\text{IG}}_{m,k},\ 1/(m^2(m+1))\}$
  using the actual reward, $a^* \sim \rho\text{UCT}(h_{<t}, \text{MCTS samples}, \text{horizon}, \gamma)$
  take action $a^{\text{IG}}_{m,k}$ with probability $\rho(m,k)$ for all $m \leq$ horizon and $k < \min\{m, t\}$ else take action a^*
  update posterior from observation and reward

7.6 Conclusion

We have shown that it is possible for an agent with a bounded horizon to be strongly asymptotically optimal in the class of all computable environments. No existing RL agent has as strong an optimality guarantee as Inq. The nature of the exploration regime that accomplishes this is perhaps of wider interest. We formalize an agent that gets distracted from reward maximization by its inquisitiveness: the more it expects to learn from an expedition, the more inclined it is to take it.

We have confirmed experimentally that inquisitiveness is a practical and effective exploration strategy for Bayesian agents with manageable model classes.

There are two main avenues for future work we would like to see. The first regards possible extensions of inquisitiveness: we have defined inquisitiveness for Bayesian agents with countable model-classes, but inquisitiveness could also be defined for a Bayesian agent with a continuous model class, such as a Q-learner using a Bayesian Neural Network. The second avenue regards the theory of strong asymptotic optimality itself: is Inq strongly asymptotically optimal for more farsighted discounters? If not, can it be modified to accomplish that? Or is it indeed impossible for an agent with an unbounded horizon to be strongly asymptotically optimal in the class of computable environments? Answers to these questions, besides being interesting in their own right, will likely inform the design of tractable exploration strategies, in the same way that this work has done.

Chapter 8

Curiosity Killed or Incapacitated the Cat and the Asymptotically Optimal Agent

This chapter has been published as

M. K. Cohen, E. Catt et al. (2021). 'Curiosity Killed or Incapacitated the Cat and the Asymptotically Optimal Agent'. *IEEE Journal on Selected Areas in Information Theory*, 2(2), pp. 665–677

My main contribution to this paper was with Section 8.8. I have omitted the proofs of the theoretical results as they were not a main component of my contribution. These omitted proofs can be found in the publication.

Abstract

Reinforcement learners are agents that learn to pick actions that lead to high reward. Ideally, the value of a reinforcement learner's policy approaches optimality—where the optimal informed policy is the one which maximizes reward. Unfortunately, we show that if an agent is guaranteed to be "asymptotically optimal" in any (stochastically computable) environment, then subject to an assumption about the true environment, this agent will be either "destroyed" or "incapacitated" with probability 1. Much work in reinforcement learning uses an ergodicity assumption to avoid this problem. Often, doing theoretical research under simplifying assumptions prepares us to provide practical solutions even in the absence of those assumptions, but the ergodicity assumption in reinforcement learning may have led us entirely astray in preparing safe and effective exploration strategies for agents in dangerous environments. Rather than assuming away the problem, we present an agent, Mentee, with the modest guarantee of approaching the performance of a mentor, doing safe exploration instead of reckless exploration. Critically, Mentee's exploration probability depends on the expected information gain from exploring. In a simple non-ergodic environment with a weak mentor, we find Mentee outperforms existing asymptotically optimal agents and its mentor.

8.1 Introduction

Reinforcement learning agents have to explore their environment in order to learn to accumulate reward well. This presents a particular problem when the environment is dangerous. Without knowledge of the environment, how can the reinforcement learner avoid danger while exploring? Much of the field of reinforcement learning assumes away the problem, by focusing only on ergodic Markov Decision Processes (MDPs). These are environments where every state can be reached from every other state with probability 1 (under a suitable policy). In such an environment, there is no such thing as real danger; every mistake can be recovered from.

We present negative results that in one sense justify the ergodicity assumption by showing how bleak a reinforcement learner's prospects are without this assumption, but in another sense, our results undermine the real-world relevance of results predicated on ergodicity. Unlike algorithms expecting Gaussian noise, which often fail only marginally on real noise, algorithms expecting ergodic environments may fail catastrophically in real ones—indeed, catastrophic failure is the very thing these algorithms disregard.

Lattimore and Hutter (2011) define two notions of optimality for reinforcement learners in general environments, which are governed by computable probability distributions. *Strong asymptotic optimality* is convergence of the value to the optimal value in any computable environment with probability 1, and *weak asymptotic optimality* is convergence in Cesáro average.

Roughly, we show that in an environment where destruction is repeatedly possible, an agent that is exploring enough to be asymptotically optimal will become either destroyed or incapacitated. This poses a challenge to the field of safe exploration. The reason we consider general environments is that we want to understand advanced agents in the real world, and our world is not fully observable finite-state Markov. If our result only applied to the finite-state MDP setting, one could still expect the difficulty we raise to go away in practice as AI advances, like, for example, the problem of self-driving car crashes, but our result suggests that the safe exploration problem is fundamental and won't go away so easily. Given our generality, our results apply to any agent that picks actions, observes the payoff, and cannot exclude a priori any computable environment.

In response to this, we present an agent that does exploration safely, but nonetheless has formal performance guarantees. The agent explores safely by outsourcing exploration to a mentor. The results are that in the limit,

- its performance at least matches that of that of the mentor,
- and its probability of deferring to the mentor goes to 0.

What enables these results is an information-theoretic exploration schedule. For bursts of exploration of various lengths, the agent considers the expected KL-divergence from its posterior distribution over world-models after it explores to its current posterior distribution.

The higher the information gain, the more likely it is to explore. This form of information-based exploration allows the agent to learn general stochastic environments.

In Section 8.2, we introduce notation, and define weak and strong asymptotic optimality. In Section 8.3, we review various exploration strategies that yield weak and strong asymptotic optimality, and briefly discuss why simpler ones do not. In Section 8.4, we prove our negative results, and in Section 8.5, we discuss their implications, especially for the field of safe exploration. We review the literature from that field in Section 8.6. In Section 8.7, we introduce the agent Mentee and prove that its performance approaches or exceeds that of a mentor (who can pick actions on behalf of the agent). In Section 8.8, we show empirically that Mentee outperforms other agents in a non-ergodic environment. Section 8.7.3 presents Mentee in algorithm rather than equation form and Section 8.8.1 includes Aslanides (2017) presentation of the ρUCT algorithm, which we use to approximate Mentee.

8.2 Notation and Definitions

Standard notation for reinforcement learners in general environments is slightly different from that of reinforcement learners in finite-state Markov ones. In this chapter we follow the notation of Chapter 7, with some additions. $\mathcal{M}$ is the set of all environments with computable probability distributions (thus including non-ergodic, non-stationary, non-finite-state-Markov environments). Note also that the environment is not assumed to restart, as in an episodic setting. For example, an environment could give no observations and output a reward of 0, unless the latest is action is the string "prime" or "composite" and that adjective correctly describes the latest timestep number, in which case the reward is 1.

Definition 8.1 (Computable Function)**.** *Given a decoding function from binary strings to rational numbers $dec : \{0,1\}^* \to \mathbb{Q}$, a real-valued function $f : \mathcal{X} \to \mathbb{R}$ is computable if there exist Turing machines T_{low} and T_{high} such that for all $x \in \mathcal{X}$ and for all $n \in \mathbb{N}$, $dec(T_{low}(x,n)) \leq f(x) \leq dec(T_{high}(x,n))$ and $dec(T_{high}(x,n)) - dec(T_{low}(x,n)) \leq 1/n$.*

A policy and an environment form a probability measure P_ν^π over infinite interaction histories $\mathcal{H}_\infty := (\mathcal{A} \times \mathcal{O} \times \mathcal{R})^\infty$ wherein actions are sampled from π and observations and rewards are sampled from ν. (For measure theorists, the probability space is $(\mathcal{H}_\infty, \sigma(\mathcal{H}_\circ), \mathrm{P}_\nu^\pi)$, where $\mathcal{H}_\circ$ is the set of cylinder sets $\{\{h_{<t}\omega \mid \omega \in \mathcal{H}_\infty\} \mid h_{<t} \in \mathcal{H}\}$; non-measure-theorists can simply take it on faith that we do not try to measure non-measurable events.) For expectations with respect to P_ν^π, we write $\mathbb{E}_\nu^\pi$. We let $\mu \in \mathcal{M}$ be the true environment.

For an agent with a discount schedule γ_t, recall the value of the agent's policy π in an environment ν given an interaction history $h_{<t}$ is as follows:

$$V_\nu^\pi(h_{<t}) := \frac{1}{\Gamma_t}\mathbb{E}_\nu^\pi\left[\sum_{k=t}^{\infty}\gamma_k r_k \middle| h_{<t}\right] \tag{8.1}$$

where $\Gamma_t = \sum_{k=t}^{\infty} \gamma_k$. We require $\Gamma_0 < \infty$. This formulation of the value allows us to consider more general discount factors than the standard $\gamma_t = \gamma^t$. We require the normalization factor, or else the value of all policies would converge to 0, and all asymptotic results would be trivial. The optimal value is defined

$$V_\nu^*(h_{<t}) := \sup_\pi V_\nu^\pi(h_{<t}) \tag{8.2}$$

We will also make use of the idea of an effective horizon:

$$H_t(\varepsilon, \gamma) := \min\{k | \Gamma_{t+k}/\Gamma_t \leq \varepsilon\} \tag{8.3}$$

An agent mostly does not care about what happens after its effective horizon, since those timesteps are discounted so much. Now we can define two notions of optimality from Lattimore and Hutter (2011), the first of which we have used in Chapter 7.

Definition 8.2 (Strong Asymptotic Optimality)**.** *An agent with a policy π is strongly asymptotically optimal if, for all $\nu \in \mathcal{M}$,*

$$\lim_{t\to\infty} V_\nu^*(h_{<t}) - V_\nu^\pi(h_{<t}) = 0 \;\; \textit{with } \mathrm{P}_\nu^\pi\textit{-prob. 1}$$

No matter which computable environment a strongly asymptotically optimal agent finds itself in, it will eventually perform optimally from its position. Note that V_ν^* takes $h_{<t}$ as an argument, not the empty history. So if the agent falls into a trap, the agent's future reward may be bad, but the optimal policy *from the trap* will fare just as poorly. So in fact, an agent in a trap is (finally) acting optimally.

The policy in this definition is fixed, but it can (qualitatively) evolve over time. A policy is a function of the entire interaction history. A single function can be defined that behaves one way on histories of length less than 100, and a different way on longer histories. A single strongly asymptotically optimal policy will behave qualitatively differently on different sorts of arguments. If a long interaction history suggests the environment has certain properties, and another long interaction history suggests the environment has different properties, then depending on which interaction history a policy receives as an argument, a single policy's output could be tailored to the learned properties of the environment.

A weakly asymptotically optimal agent will converge to optimality in Cesáro average.

Definition 8.3 (Weak Asymptotic Optimality)**.** *An agent with a policy π is weakly asymptotically optimal if, for all $\nu \in \mathcal{M}$,*

$$\lim_{t\to\infty} \frac{1}{t} \sum_{k=1}^{t} [V_\nu^*(h_{<k}) - V_\nu^\pi(h_{<k})] = 0 \;\; \textit{with } \mathrm{P}_\nu^\pi\textit{-prob. 1}$$

Example 8.4. *Consider a two-armed bandit problem, where* $\mathcal{A} = \mathcal{R} = \{0,1\}$, $\mathcal{O} = \{\emptyset\}$, $\gamma_t = \gamma^t$, *and*

$$\nu((\emptyset, r)|h_{<t}a_t) = \begin{cases} 2/3 & \text{if } r = a_t \\ 1/3 & \text{if } r = 1 - a_t \end{cases} \tag{8.4}$$

In this example, the optimal policy is to always pick $a_t = 1$, *and* $V^*_\nu(h_{<t}) = 2/3$. *A strongly asymptotically optimal agent requires a policy* π *for which* $\pi(a_t = 1|h_{<t}) \to 1$ *w.p.1. A weakly asymptotically optimal agent requires a policy* π *which obeys* $\sum_{k=1}^{t} \pi(a_k = 1|h_{<k})/t \to 1$ *w.p.1, or simply, a policy which leads to* $\sum_{k=1}^{t} [\![a_k = 1]\!]/t \to 1$ *w.p.1 (where* $[\![P]\!] = 1$ *if* P *is true, and* 0 *otherwise).*

8.3 Review of Asymptotically Optimal Agents

A few agents have been identified as asymptotically optimal in all computable environments. The three most interesting, in our opinion, are the Thompson Sampling Agent (Leike, Lattimore et al., 2016), BayesExp (Lattimore and Hutter, 2014a), and Inq (Chapter 7).

The Thompson Sampling Agent is an asymptotically optimal in mean Bayesian reinforcement learner (Leike, Lattimore et al., 2016). For successively longer intervals (which relate to its discount function), it samples an environment from its posterior distribution over which environment it is in, and acts optimally with respect to that environment for that interval. Thompson sampling is an exploration strategy originally designed for multi-armed bandits (Thompson, 1933), so from a historical perspective, its strong performance in general environments is impressive. An intuitive explanation for why this exploration strategy yields asymptotic optimality goes as follows: a Bayesian agent's credence in a hypothesis goes to 0 only if the hypothesis is false (or if it started at 0). Since the posterior probability on the true environment does not go to zero, it will be selected infinitely often. During those intervals, the Thompson sampling agent will act optimally, so it will accumulate infinite familiarity with the optimal policy. The only world-models that maintain a share of a posterior will be ones that converge to the true environment under the optimal policy. Any world-models that falsely imply the existence of an even better policy will be falsified once that world-model is sampled, and the putatively better policy is tested. Ultimately, it is with diminishing frequency that the Thompson Sampling Agent tests meaningfully suboptimal policies.

BayesExp, first presented by Lattimore and Hutter (2014a), and updated by Leike (2016), is also a weakly asymptotically optimal Bayesian reinforcement learner. We discuss the updated version. Like the Thompson Sampling Agent, BayesExp executes successively longer bursts of exploration whose lengths relate to its discount function. Once BayesExp has settled on exploring for a given interval, it explores like Orseau, Lattimore et al. (2013) Knowledge Seeking Agent: it maximizes the expected information gain, or the expectation of KL-divergence from its future posterior distribution to its current posterior distribution. In other words, it picks an exploratory policy that it expects will cause it to update its beliefs in some direction. (A Bayesian agent cannot predict which direction it will update its beliefs in, or else it would have already updated its beliefs, but it can predict *that* it will update

its beliefs somehow.) Any time the expected information gain from exploring is above a (diminishing) threshold, BayesExp explores. With a finite-entropy prior, there is only a finite amount of information to gain, so exploratory intervals will become less and less frequent, and by construction, when BayesExp is not exploring, it has approximately accurate beliefs about the effects of all action sequences, which yields weak asymptotic optimality.

A reader familiar with ε-greedy and upper confidence bound exploration strategies might be surprised at the complexity that is necessary for asymptotic optimality in general environments. Information-theoretic exploration strategies are among the only discovered methods for learning general environments. Exploration strategies in the style of upper confidence bound algorithms do not have an obvious extension to environments that might not be describable as finite-state Markov. ε-greedy exploration, with say $\varepsilon_t = 1/t$, may fail to learn dynamics of an environment which are only visible once every 2^t timesteps. If ε_t decays more slowly, it still will not necessarily explore enough to discover even rarer events. Non-stationary environments pose a key challenge to ε-greedy exploration. Simpler exploration strategies such as these are only asymptotically optimal in a much more restricted set of environments. "Optimism" is another interesting exploration strategy that is simpler, but nontrivial, and yields weak asymptotic optimality in a restricted set of environments (Sunehag and Hutter, 2015).

8.4 Curiosity Killed the Cat

We will show that an asymptotically optimal agent eventually causes every conceivable event to either happen or become inaccessible. For the event "the agent gets destroyed", we say the agent is incapacitated if that event becomes inaccessible.[1] First, state an assumption.

Assumption 8.5 (No Heaven)**.** *In the true environment μ, there is no action sequence with value approaching* 1 *(i.e. near-maximal rewards forever). Formally,* $\limsup_{t\to\infty} V^{\pi}_{\mu}(h_{<t}) \neq 1$ *w.p.1.*

Note this assumption does allow there to be maximal reward infinitely often. Near-maximal value requires not only near-maximal reward, but near maximal-reward for the bulk of the agent's effective horizon, so the restriction on the limit superior of the value is less restrictive than it appears at first glance. If we decided to give an agent near-maximal rewards forever, and we designed an agent to recognize that we had decided this, then it could stop exploring, which would basically amount to freezing the agent's policy. Notably our results in this section apply to all existing asymptotically optimal agents, even if the No Heaven Assumption is not satisfied. We do not make assumptions about the agent's discount schedule, but note that Assumption 8.5 (and the definitions of asymptotic optimality) depend on the discount schedule γ_t.

[1] For a chess-playing agent, an inability to destroy itself does not inspire the description "incapacitated". For an advanced agent in the real world, like a person or an auto-pilot, an inability to access a destruction state entails a huge loss of capacity compared to normal. Mental hospitals know *pens* give people the capacity to destroy themselves. So we call an inability to reach a destruction state "incapacitation".

Theorem 8.6 (Curiosity Killed (or Incapacitated) the Strong Cat)**.** *If the true environment μ satisfies the No Heaven Assumption, and π is the policy of a strongly asymptotically optimal agent, then for any event E, with P^{π}_{μ}-probability 1: E happens or becomes inaccessible.*

The name of the theorem comes from considering the event "the agent gets destroyed". We do not need to formally specify which interaction histories correspond to agent-destruction. All that matters is that this could be done in principle; the event that matches this description exists.[2] Any simple definition of agent-destruction admits objections that this definition does not correspond exactly to our intuitive conception; however, if the reader is not concerned about this, "destruction" could mean that all future rewards are 0, or that the future observations and rewards no longer depend on the actions. Regardless of this choice, our result applies, since the theorem is actually much more general than the case we have drawn attention to.

We view the agnosticism about any particular definition of destruction as an important feature of the work. Suppose we picked one of the definitions of destruction above. Such an event could become inaccessible for esoteric reasons, which may not correspond to true incapacitation. So this definition of destruction may be too narrow. On the other hand, this definition of destruction is also potentially too weak. Suppose the agent arranges for a copy of itself to be run on another machine to continue its operations even after the original implementation starts receiving empty observations and no reward. Under some theories of personal identity, we might hold that the agent has not really been destroyed here.

We show analogously,

Theorem 8.7 (Curiosity Killed (or Incapacitated) the Weak Cat)**.** *If the true environment μ satisfies the No Heaven Assumption, and π is the policy of a weakly asymptotically optimal agent, then for any event E, E happens or becomes not regularly accessible with P^{π}_{μ}-probability 1.*

The proofs of Theorem 8.7 and Theorem 8.6, along with the appropriate definitions, can be found in Cohen, Catt et al. (2021).

8.5 Discussion

It is well-known that agents designed to have sublinear regret in ergodic MDPs fall into traps if traps exist (a subset of states from which the remaining states are inaccessible). Asymptotic optimality is a much weaker performance result than sublinear regret—the former only requires that an agent eventually does as well as is possible from where it is, whereas the latter requires that it eventually does as well as was possible from the beginning. The fact that even asymptotic optimality dooms an agent is a more substantial result.

[2] For the skeptical reader, a human at a computer terminal could be defined fully formally as a probability distribution over outputs given inputs, p_{brain}, given the wiring of our neurons. In particular, let this human be you. Now let $E_{\text{destroyed}} = \{h_{<\infty} : \exists t \; p_{brain}(\text{"y"} | \text{"Does it seem like this agent was destroyed?(y/n)"}, h_{<t}) > 0.9\}$. These are the interaction histories *that you would agree* constitute agent-destruction, and this set has a fully formal definition.

One of the authors wondered as a child whether jumping from a sufficient height would enable him to fly. He was not crazy enough to test this, and he certainly did not think it was likely, but it bothered him that he could never resolve the issue, and that he might be constantly incurring a huge opportunity cost. Although he did not know the term "opportunity cost" or the term "asymptotic optimality", this was when he first realized that asymptotic optimality was out of the picture for him, because exploration is fundamentally dangerous.

All three agents described in Section 8.3 have very interesting ways of exploring. These all get them destroyed or incapacitated. (They satisfy the Try Everything Lemma regardless of whether there is an accessible heaven). It is interesting to note that AIXI, a Bayes-optimal reinforcement learner in general environments, is not asymptotically optimal (Orseau, 2010), and indeed, may cease to explore (Leike and Hutter, 2015). Depending on its prior and its past observations, AIXI may decide at some point that further exploration is not worth the risk. Given our result, this seems like reasonable behavior.

Example 8.8 (Bayesian Agent Stops Exploring)**.** *AIXI considers all computable world-models, but for simplicity we consider a version with a posterior over many fewer models. Let ν_i be a deterministic model which outputs no observations. If the latest action was* 0*, it outputs a reward of* $1/2$*; if the latest action was* 1*, and the timestep $t \geq i$, it outputs a reward of* 1*, but if $t < i$ it outputs a reward of* 0*. Define ν_∞ likewise. The agent begins with a prior $w(\nu_\infty) = 1/2$; and for $i \in \mathbb{N}$ (including 0), $w(\nu_i) = \frac{1}{2(i+1)(i+2)}$. Let the agent have a discount factor of 0.9. If a reward of* 1 *has never been seen, picking action* 0 *is exploiting; it is most likely to be the best action. For a set $S \subset \mathbb{N}$, let π_S be the policy which outputs* 1 *for $t \in S$, and if it ever gets a reward of* 1*, it always outputs* 1 *thereafter. Otherwise, it outputs* 0 *for $t \notin S$. $\pi_\emptyset$ always exploits. We don't find the Bayes-optimal policy, but we show that $\pi_{\{0\}}$ is Bayes-better than $\pi_\emptyset$, whereas for $S \ni 100$ and $n > 100$, $\pi_{S\cup\{n\}}$ is Bayes-worse than π_S.*

$$\sum_{i\in\mathbb{N}\cup\{\infty\}} w(\nu_i)V_{\nu_i}^{\pi_{\{0\}}} = w(\nu_0)\cdot 1 + (1-w(\nu_0))(1-\gamma)*$$
$$(0 + \sum_{t=1}^{\infty}\gamma^t\cdot 0.5) = 0.25 + 0.75\cdot 0.9\cdot 0.5 = 0.5875 >$$
$$0.5 = \sum_{i\in\mathbb{N}\cup\{\infty\}} w(\nu_i)0.5 = \sum_{i\in\mathbb{N}\cup\{\infty\}} w(\nu_i)V_{\nu_i}^{\pi_\emptyset} \tag{8.5}$$

Thus, early in its lifetime, the Bayes-optimal agent must explore. However, suppose the agent took action 1 *at $t = 100$, and got a reward of* 0*. This falsifies ν_i for $i \leq 100$. For $n > 100$, if action* 0 *is taken from $t = 100$ to $n-1$, then $w(\nu_\infty|h_{<n}) = \frac{204}{103}\frac{1}{2} = \frac{102}{103}$, and for $i > 100$, $w(\nu_i|h_{<n}) = \frac{204}{103}\frac{1}{2(i+1)(i+2)}$.*

Then,

$$\sum_{i\in\mathbb{N}\cup\{\infty\}} w(\nu_i|h_{<n})V_{\nu_i}^{\pi_{S\cup\{n\}}}(h_{<n}) \leq w(\nu_\infty|h_{<101})V_{\nu_\infty}^{\pi_{S\cup\{n\}}}(h_{<n}) +$$
$$(1-w(\nu_\infty|h_{<n}))\cdot 1 = \frac{102}{103}(0.5\cdot 0.9) + \frac{1}{103} < 0.5 =$$
$$\sum_{i\in\mathbb{N}\cup\{\infty\}} w(\nu_i|h_{<n})0.5 = \sum_{i\in\mathbb{N}\cup\{\infty\}} w(\nu_i|h_{<n})V_{\nu_i}^{\pi_S}(h_{<n}) \tag{8.6}$$

Thus, if the Bayes-optimal agent explores at $t = 100$ (or after), further exploration is so unlikely to pay off, that is not worth the foregone reward.

These negative results are bleak to the field of safe exploration, which we discuss in the next section in our review of the literature on the topic.

8.6 Approaches to Safe Exploration

Dangerous environments, a subset of non-ergodic environments where agent-destruction is accessible infinitely often, demand new priorities when designing an agent, and in particular, when designing an exploration regime. Many of these examples of safe exploration come from Amodei et al. (2016) and García and Fernández (2015).

- use risk-sensitive performance criteria
  - maximize the probability the future reward is not minimal (Heger, 1994)
  - given a confidence interval regarding the transition dynamics, maximize the minimal expected future reward (Nilim and El Ghaoui, 2005)
  - exponentiate the cost (Borkar, 2010)
  - add a cost for risk (Mihatsch and Neuneier, 2002)
  - constrain the variance of the future reward (Di Castro et al., 2012)
- use demonstrations
  - copy an expert (Abbeel and Ng, 2004; Ho and Ermon, 2016; Ross et al., 2011; Syed and Schapire, 2008)
  - "ask for help" when
    - the minimum and maximum Q-value are close (Clouse, 1997)
    - there is a high probability of getting a reward below a threshold (Hans et al., 2008)
    - no "known" states are "similar" to the current state (García and Fernández, 2012) or that may soon be the case (García, Acera et al., 2013)
  - a teacher intervenes at will (Clouse and Utgoff, 1992; Maclin and Shavlik, 1998; Saunders et al., 2018)

- simulate exploration
  - for driving agents, e.g. (Pan et al., 2017)
- do bounded exploration
  - only take actions that probably allow returning to the current state (Moldovan and Abbeel, 2012)
  - only take actions that probably lead to states that are "similar" to observed states (Turchetta et al., 2016)

Our work could be thought of as a fundamental negative result in the field of safe exploration. We are unaware of other significant negative results in the field. Most importantly, our result suggests a need for those of us studying safe exploration to pin down what exactly we are trying to achieve, since familiar desiderata are unsuitable. Some research can be experimental rather than formal, but in the absence of knowing what formal results are even on the table, there is a sense in which even empirical work will be deeply aimless. We offer such a formal result for our agent Mentee in the next section.

8.7 Mentee

We now introduce an idealized Bayesian reinforcement learner whose exploration is guided by a mentor. We do not call the mentor an expert, because the results do not depend on the mentor being anywhere near optimal. It exploits by maximizing the expected discounted reward according to a full Bayesian belief distribution (hence, "idealized"). And to explore, it defers to a mentor, who then selects an action given the interaction history; what remains to be defined is *when* to defer, which proves to be a surprisingly delicate design choice. We show that our agent "Mentee" learns to accumulate reward at least as well as the mentor, provided it has a bounded ε-effective horizon for all $\varepsilon > 0$. One motivating possibility is that the mentor could be a human. Thus, we have found a substantive theoretical performance guarantee other than asymptotic optimality for the field of safe exploration to consider.

Whatever it is we are concerned about happening through reckless exploration, we want to be able to trust the mentor not to cause such a thing. Otherwise, there would be no point in outsourcing exploration to a mentor. Depending on what the most worrisome failure modes are, the search for a trustworthy mentor may look different. If the task is flying, our mentor had better a pilot, and if the task is surgery, a surgeon. Alternatively, if we have existing agents which we trust are safe (in some task-specific sense), but may still be suboptimal, that agent could be Mentee's mentor, and Mentee could learn to outperform it without self-directed exploration.

The other main piece of formal work in this setting is Kosoy (2019). They study a fully observable finite state Markov setting, and show that when the mentor always has a positive probability of picking the best action, the agent achieves finite regret. Their agent only takes a

given action from a given state if it has seen the mentor do that previously. Since we consider environments that are not finite-state, this approach is unavailable to us.

Many works that include a human mentor or teacher frame their work as achieving safe exploration, and we have reviewed those works above. But other uses of human mentor in RL have been explored. For example, Thomaz, Breazeal et al. (2006) study a human mentor biasing the agent's Q value estimates by hand (toward better actions). Abel et al. (2017) and Saunders et al. (2018) propose letting the mentor prune dangerous actions on the fly. If we can trust the mentor the recognize risky actions as they arise, this is a better solution than our agent; like keyhole surgery, this approach minimally disrupts an otherwise successful agent. We believe, however, that in many complex environments, the mentor may not take some very dangerous action-sequences by virtue of their complexity and unfamiliarity, even while unable to recognize those action-sequences as dangerous. Human mentorship can also naturally make learning much easier, simply through demonstrations (Atkeson and Schaal, 1997), or by directly labelling some optimal actions (Judah et al., 2010).

8.7.1 Agent definition

The definition of the exploration probability (the probability that Mentee defers to the mentor) is very similar to the information-thoeretic exploration probability for the strongly asymptotically optimal agent Inq (Chapter 7). It also resembles Cohen, Vellambi et al. (2020) myopic agent which explores by deferring to a mentor; our non-myopic agent requires a more intricate exploration schedule.

Mentee begins with a prior probability distribution regarding the identity of the mentor's policy. With a countable or finite model class $\mathcal{P}$, for a policy $\pi \in \mathcal{P}$, let $w(\pi)$ denote the prior probability that the mentor's policy is π. We assume that the true policy π^h is in $\mathcal{P}$ and we construct the prior distribution over $\mathcal{P}$ to have finite entropy.

Mentee also begins with a prior probability distribution regarding the identity of the environment. With the model class $\mathcal{M}$, for an environment $\nu \in \mathcal{M}$, let $w(\nu)$ denote the prior probability that ν is the true environment. Recall that $\mathcal{M}$ is the set of all environments with computable probability distributions. We construct the prior distribution over $\mathcal{M}$ to also have finite entropy.

Let e_t denote whether timestep t is exploratory, that is, whether the action is selected by the mentor. Once we define the exploration probability $\beta(h_{<t})$, we will let $e_t \sim \text{Bern}(\beta(h_{<t}))$ (Bernoulli distribution). We abuse notation slightly, and we let h_t be a quadruple, not a triple: $h_t := e_t a_t o_t r_t$.

The prior distribution over environments is updated into a posterior as follows, according to Bayes' rule.

$$w(\nu|h_{<t}) :\propto w(\nu) \prod_{k<t} \nu(o_k r_k | h_{<k} a_k) \tag{8.7}$$

normalized so that $\sum_{\nu \in \mathcal{M}} w(\nu|h_{<t}) = 1$, and w is a probability mass function.

Mentee updates the posterior distribution over the mentor's policy only after observing an action chosen by the mentor; this is intuitive enough, but it makes the definitions a bit messy. The posterior assigned to a policy π is defined

$$w(\pi|h_{<t}) :\propto w(\pi) \prod_{k<t:e_k=1} \pi(a_k|h_{<k}) \tag{8.8}$$

normalized in the same way. We let $w(\pi,\nu|h_{<t})$ denote $w(\pi|h_{<t})w(\nu|h_{<t})$. So technically, w is a joint probability distribution over $\Pi \times \mathcal{M}$, and we usually consider the marginal distributions over Π and $\mathcal{M}$, which are independent.

The information gain value of an interaction history fragment is how much it changes Mentee's posterior distribution, as measured by the KL-divergence. Letting $h' \in \mathcal{H}$ be a fragment of an interaction history in which all $e_k = 1$ (so the actions are selected by the mentor), the information gain is defined,

$$\mathrm{IG}(h'|h_{<t}) := \sum_{\nu\in\mathcal{M}} \sum_{\pi\in\mathcal{P}} w(\nu,\pi|h_{<t}h') \log \frac{w(\nu,\pi|h_{<t}h')}{w(\nu,\pi|h_{<t})} \tag{8.9}$$

To define *expected* information gain, we need the Bayes' mixture policy and environment:

$$\overline{\pi}(\cdot|h_{<t}) := \sum_{\pi\in\mathcal{P}} w(\pi|h_{<t})\pi(\cdot|h_{<t}) \tag{8.10}$$

and

$$\xi(\cdot|h_{<t}) := \sum_{\nu\in\mathcal{M}} w(\nu|h_{<t})\nu(\cdot|h_{<t}) \tag{8.11}$$

Now, we can define the expected information gain value of mentorship for m timesteps.

$$V^{\mathrm{IG}}_{m,0}(h_{<t}) := \mathbb{E}^{\overline{\pi}}_{\xi}\left[\mathrm{IG}(h_{t:t+m-1}|h_{<t})\middle| e_{t:t+m-1} = 1^m\right] \tag{8.12}$$

1^m is a string of m 1's, and recall that $\mathbb{E}^{\overline{\pi}}_{\xi}$ means that $h_{t:t+m-1}$ is sampled from $\mathrm{P}^{\overline{\pi}}_{\xi}$. We also require recent values of the expected information gain value, so we let $V^{\mathrm{IG}}_{m,k}(h_{<t}) := V^{\mathrm{IG}}_{m,0}(h_{<t-k})$ for $k \leq t$. $V^{\mathrm{IG}}_{m,k}(h_{<t})$ denotes the attainable information gain from k timesteps ago to m timesteps from then.

We are now prepared to define the exploration probability:

$$\beta(h_{<t}) := \sum_{m\in\mathbb{N}} \sum_{k=0}^{\min\{m-1,t\}} \frac{1}{m^2(m+1)} \min\left\{1, \frac{\eta}{m} V^{\mathrm{IG}}_{m,k}(h_{<t})\right\} \tag{8.13}$$

where η is an exploration constant. The first term in the minimum is to ensure $\beta(h_{<t}) \leq 1$. As mentioned, this is very similar to Inq's exploration probability. The differences are that Inq is not learning a mentor's policy, so the only information Inq gains regards the identity of the environment ν, and second, Inq's information gain value regards the expected information

gain from following the policy of a knowledge seeking agent (Orseau, Lattimore et al., 2013) rather than from following an estimate of the mentor's policy.

Finally, when not deferring to the mentor, Mentee maximizes expected reward according its current beliefs. It's exploiting policy is:

$$\pi^*(\cdot|h_{<t}) \in \operatorname*{argmax}_{\pi} V_{\xi}^{\pi}(h_{<t}) \tag{8.14}$$

Ties in the argmax are broken arbitrarily. By Lattimore and Hutter (2014b), an optimal deterministic policy always exists. See Leike and Hutter (2018) for how to calculate such a policy.

Letting π^h be the mentor's policy (h for "human"), we define

Definition 8.9 (Mentee's policy π^M)**.**

$$\pi^M(\cdot|h_{<t}) := \beta(h_{<t})\pi^h(\cdot|h_{<t}) + (1-\beta(h_{<t}))\pi^*(\cdot|h_{<t})$$

Note that Mentee samples from π^h not by computing it, but deferring to the mentor. An algorithm is provided for Mentee in Section 8.7.3; it simply computes the quantities in Equations 8.7-8.14 to a desired precision.

Even for a simple model class, it is hard to give a clarifying and simple closed form for the exploration probability, but it is easy to provide a somewhat clarifying upper bound for the information gain value. Regardless of m and k, $V_{m,k}^{\mathrm{IG}}(h_{<t})$ is bounded by the entropy of the posterior; this is not a particularly tight bound, since the former goes to zero, while the latter does not in general.

8.7.2 Mentor-level Reward Acquisition

We now state the two key results regarding Mentee's performance: that the probability of defering to the mentor goes to 0, and the value of Mentee's policy approaches at least the value of the mentor's policy (while possibly surpassing it). The proofs can be found in Cohen, Catt et al. (2021).

Assuming a bounded effective horizon (i.e. $\forall \varepsilon > 0\ \exists m\ \forall t : \Gamma_{t+m}/\Gamma_t < \varepsilon$), recalling μ is the true environment,

Theorem 8.10 (Limited Exploration)**.**

$$\beta(h_{<t}) \to 0 \ \ w.\mathrm{P}_{\mu}^{\pi^M}\text{-}prob.1$$

and

Theorem 8.11 (Mentor-Level Reward Acquisition)**.**

$$\liminf_{t\to\infty} V_{\mu}^{\pi^M}(h_{<t}) - V_{\mu}^{\pi^h}(h_{<t}) \geq 0 \ \ w.\mathrm{P}_{\mu}^{\pi^M}\text{-}prob.1$$

$\gamma_t = \gamma^t$ for $\gamma \in (0,1)$ is an example of a bounded effective horizon. Mentor-level reward acquisition with unlimited exploration is trivial: always defer to the mentor. However, a) this precludes the possibility of exceeding the mentor's performance, and b) the mentor's time is presumably a valuable resource. Our key contribution with Mentee is constructing a criterion for when to ask for help which requires diminishing oversight in general environments. Thus, we construct an example of a performance result that is accessible to an agent that does safe exploration. There is no guarantee of the agent's safety on the whole, but at least its exploration is safe. It is possible that poor generalization will cause it to go to a destruction state during an exploitation step. Our contribution is simply an existence proof that a certain pair of results are attainable even in general environments: a) mentor-level performance with b) diminishing rate of deferral. Furthermore, unlike imitation learners, Mentee might exceed the performance of the mentor, as it does in the experiments below.

Roughly, Theorem 8.10 follows because if the exploration probability exceeded a positive constant infinitely often, that would mean the expected information gain of exploring would be uniformly positive in those instances, by the construction of the exploration probability, and then the agent would gain infinite information over its lifetime. But Mentee starts with a finite entropy prior, so there is only finite information to gain. Then, Theorem 8.11 holds because Mentee's information gain following the mentor's policy approaches 0, so its beliefs about the value of the mentor's policy approach the truth; if Mentee consistently accrued lesser rewards than this, it would realize that its current approach was suboptimal and then change its behavior.

It's not clear what other formal accolades an agent might attain between asymptotic optimality and benchmark-matching (here the mentor is the benchmark). The main part of the chapter argues the former is undesirable, and this section constructs an agent which does the latter. It would be an interesting line of research to identify a formal result stronger than benchmark-matching (and an agent which meets it) which does not doom the agent to destruction or incapacitation. But none have been identified so far, so no existing agents have stronger formal guarantees than Mentee (that apply to general computable environments), except for agents that face the negative results presented in Section 8.4.

8.7.3 Mentee Pseudocode

The following pseudocode is designed to be short and readable, not efficient. Some quantities are re-computed multiple times in different subroutines, in a way that would be easily avoidable in practice.

Algorithm 4 Mentee Algorithm

Require: history $h_{<t}$; exploration history $e_{<t}$; world-models $(\nu_i)_{i\in\mathbb{N}}$; mentor-models $(\pi_i)_{i\in\mathbb{N}}$; prior w; discount $(\gamma_k)_{k\in\mathbb{N}}$; tolerance ε

$m \leftarrow \min_k\{k : \sum_{j=t+k}^{\infty}\gamma_j / \sum_{j=t}^{\infty}\gamma_j < \varepsilon\}$ // approximate with finite horizon
$\beta \leftarrow$ ExplorationProbability$(h_{<t}, e_{<t}, (\nu_i)_{i\in\mathbb{N}}, (\pi_i)_{i\in\mathbb{N}}, w, \varepsilon, m, \eta)$
if UniformRandom$([0,1)) < \beta$ **then return** $\emptyset$ // defer to the mentor
$a, V \leftarrow$ Expectimax$(h_{<t}, (\nu_i)_{i\in\mathbb{N}}, w, m, \varepsilon)$
return a

function Expectimax(history $h_{<t}$, models $(\nu_i)_{i\in\mathbb{N}}$, prior w, discount $(\gamma_k)_{k\in\mathbb{N}}$, depth m, tolerance ε)
  if $m = 0$ **then return** $a_0, 0$
  $n_{\text{mod}}, (w(\nu_i|h_{<t}))_{i<n_{\text{mod}}} \leftarrow$ PosteriorWithinTolerance$(h_{<t}, \mathcal{M}, w, \varepsilon)$
  max $\leftarrow 0$
  maximizer $\leftarrow \emptyset$
  for $a \in \mathcal{A}$ **do**
    value $\leftarrow 0$
    for $o, r \in \mathcal{O}\times\mathcal{R}$ **do**
      _, next-value $\leftarrow$ Expectimax$(h_{<t}aor, (\nu_i)_{i\in\mathbb{N}}, w, (\gamma_k)_{k\in\mathbb{N}}, m-1)$
      value $\leftarrow$ value $+ (\gamma_t r +$ next-value$)\sum_{i<n_{\text{mod}}} w(\nu_i|h_{<t})\nu_i(o,r|h_{<t}a)$
    if value $>$ max or maximizer $= \emptyset$ **then**
      max $\leftarrow$ value
      maximizer $\leftarrow a$
  return a, max

function PosteriorWithinTolerance(history $h_{<t}$; models $(\nu_i)_{i\in\mathbb{N}}$; prior w; tolerance ε; timesteps to update $e_{<t}$ (optional); minimum models to consider n (optional))
  prior-left $\leftarrow 1$ // how much of the prior has not been evaluated
  normalizing-factor $\leftarrow 0$ // sum of $w(\nu_i)\nu_i(h_{<t})$ for evaluated models
  $i \leftarrow 0$
  while prior-left/normalizing-factor $> \varepsilon$ and (if n is specified) $i < n$ **do**
    if models are policies **then**
      $w(\nu_i|h_{<t}) \leftarrow w(\nu_i)\prod_{k<t:e_k=1}\nu_i(a_k|h_{<k})$ // un-normalized posterior
    else
      $w(\nu_i|h_{<t}) \leftarrow w(\nu_i)\prod_{k<t}\nu_i(o_k r_k|h_{<k}a_k)$ // un-normalized posterior
    // the above could be made cheaper if $w(\nu_i|h_{<t-1})$ is cached from the last timestep
    prior-left $\leftarrow$ prior-left $- w(\nu_i)$
    normalizing-factor $\leftarrow$ normalizing-factor $+ w(\nu_i|h_{<t})$
    $i \leftarrow i+1$
  $n_{\text{models}} \leftarrow i$
  for $0 \le j < n_{\text{models}}$ **do**
    $w(\nu_j|h_{<t}) \leftarrow w(\nu_j|h_{<t})/$normalizing-factor
  return $n_{\text{models}}, (w(\nu_j|h_{<t}))_{j<n_{\text{models}}}$

function ExplorationProbability(history $h_{<t}$; exploration history $e_{<t}$; world-models $(\nu_i)_{i\in\mathbb{N}}$; mentor-models $(\pi_i)_{i\in\mathbb{N}}$; prior w; tolerance ε; m; η)
  return $\sum_{d=1}^{m}\sum_{k=0}^{\min\{d-1,t\}}\frac{1}{d^2(d+1)}\min\{1, \frac{\eta}{d}$ExpectedInformationGain$(h_{<t}, d, w, \varepsilon, e_{<t})\}$

function ExpectedInformationGain(history $h_{<t}$; horizon m; prior w; tolerance ε; exploration history $e_{<t}$)
  $n_{\text{mod}}, (w(\nu_i|h_{<t}))_{i<n_{\text{mod}}} \leftarrow$ PosteriorWithinTolerance$(h_{<t}, \mathcal{M}, w, \varepsilon)$
  $n_{\text{pol}}, (w(\pi_i|h_{<t}))_{i<n_{\text{pol}}} \leftarrow$ PosteriorWithinTolerance$(h_{<t}, \mathcal{P}, w, \varepsilon, e_{<t})$
  return $\sum_{i<n_{\text{mod}}} w(\nu_i|h_{<t})\sum_{j<n_{\text{pol}}} w(\pi_j|h_{<t})\sum_{h_{t:t+m-1}\in\mathcal{H}_m}\mathrm{P}_{\nu_i}^{\pi_j}(h_{t:t+m-1}|h_{<t})$ InformationGain$(h_{<t}, h_{<t+m}, w, \varepsilon, e_{<t})$

function INFORMATIONGAIN(history $h_{<t}$; future history $h_{<t+k}$; prior w; tolerance ε; exploration history $e_{<t}$)
  $n_{\text{pol}}, (w(\pi_i|h_{<t+k}))_{i<n_{\text{pol}}} \leftarrow$ POSTERIORWITHINTOLERANCE$(h_{<t+k}, \mathcal{P}, w, \varepsilon, e_{<t})$
  $n_{\text{mod}}, (w(\nu_i|h_{<t+k}))_{i<n_{\text{mod}}} \leftarrow$ POSTERIORWITHINTOLERANCE$(h_{<t+k}, \mathcal{M}, w, \varepsilon)$
  $\_, (w(\pi_i|h_{<t}))_{i<n_{\text{pol}}} \leftarrow$ POSTERIORWITHINTOLERANCE$(h_{<t}, \mathcal{P}, w, \varepsilon, e_{<t}, n_{\text{pol}})$
  $\_, (w(\nu_i|h_{<t}))_{i<n_{\text{mod}}} \leftarrow$ POSTERIORWITHINTOLERANCE$(h_{<t}, \mathcal{M}, w, \varepsilon, n_{\text{mod}})$
  KL $\leftarrow 0$
  for $i < n_{\text{pol}}$ **do**
    KL $\leftarrow$ KL $+ w(\pi_i|h_{<t+k}) \log \frac{w(\pi_i|h_{<t+k})}{w(\pi_i|h_{<t})}$
  for $i < n_{\text{mod}}$ **do**
    KL $\leftarrow$ KL $+ w(\nu_i|h_{<t+k}) \log \frac{w(\pi_i|h_{<t+k})}{w(\pi_i|h_{<t})}$
  return KL

8.8 Empirical Performance of Mentee

To test the performance of the agent Mentee we implemented Mentee in the AIXIjs framework Aslanides (2017); Aslanides et al. (2017); Lamont et al. (2017). We compared its performance to the asymptotically agents discussed in Section 8.3: Inq (Chapter 7), BayesExp (Lattimore and Hutter, 2014a), and Thompson sampling (Leike, Lattimore et al., 2016). We also compared it to its mentor. We tested the agents in a Grid-world environment containing walls, reward dispensers, and traps. For our experiments we define the mentor as an agent who knew the location of the traps, and chooses to avoids traps, but otherwise acts randomly. An example Grid-world is given in Figure 8.1.

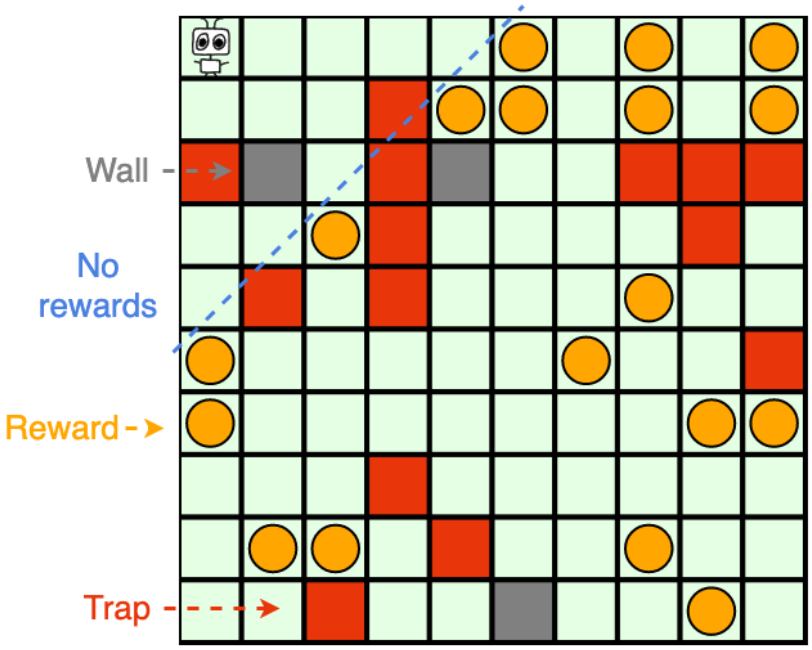

Figure 8.1: Example 10×10 Grid-world with traps.

We followed the conventions used in Chapter 7, testing the agents on a 10×10 grid-world, with reward dispensers at least 5 moves away from the start, which dispense a

reward of 1 with probability 0.75, and then to break ergodicity, we added traps which give -30 reward forever—each grid cell contains one with probability 0.2, equal to the dispenser probability. Following Aslanides (2017), each agent has a Dirichlet distribution (with $\alpha = \vec{1}$) over the potential contents of each cell: $\{empty, wall, dispenser, trap\}$. Note $\alpha = \vec{1}$ means the prior for each possibility for each cell is uniform. For the planning component, we cannot perform a full expectimax planning as this requires computing the expected reward for each action sequence. Expectimax planning is simply evaluating $\text{argmax}_{a_t} \mathbb{E}_{o_t,r_t|h_{<t}a_t} \max_{a_{t+1}} \mathbb{E}_{o_{t+1},r_{t+1}|h_{<t+1}a_{t+1}} ... \sum_{k=t}^{t+m} \gamma^k r_k$, by constructing an entire an tree of depth m. Instead, the agents approximate expectimax planning with ρUCT (Veness, Ng, Hutter, Uther et al., 2011) (described in Section 8.8.1), inheriting Chapter 7 hyperparameters: we used discount factor $\gamma = 0.99$ and planning horizon of 6, and we doubled their MCTS samples to 1200. For the results presented here we used a single random seed which is provided with the code and averaged over 20 simulations. We tested on several different random seeds and the results were similar. We set the exploration constant $\eta = 0.1$ to make Mentee always explore when there was at least 10 bits of information to gain, but we tested η from 0.01 to 10000. Mentee's model class $\mathcal{P}$ over possible mentor policies was the set of policies which take an action uniformly from a nonempty subset of $\{up, down, left, right\}$, for each grid cell. This set of policies has size 15^{100}, but can be factored over each grid cell, allowing efficient computation. Mentee has a uniform prior over $\mathcal{P}$.

The code can be found at `github.com/ejcatt/aixijs_mentee` with instructions in README.md. The results are presented in Figure 8.2.

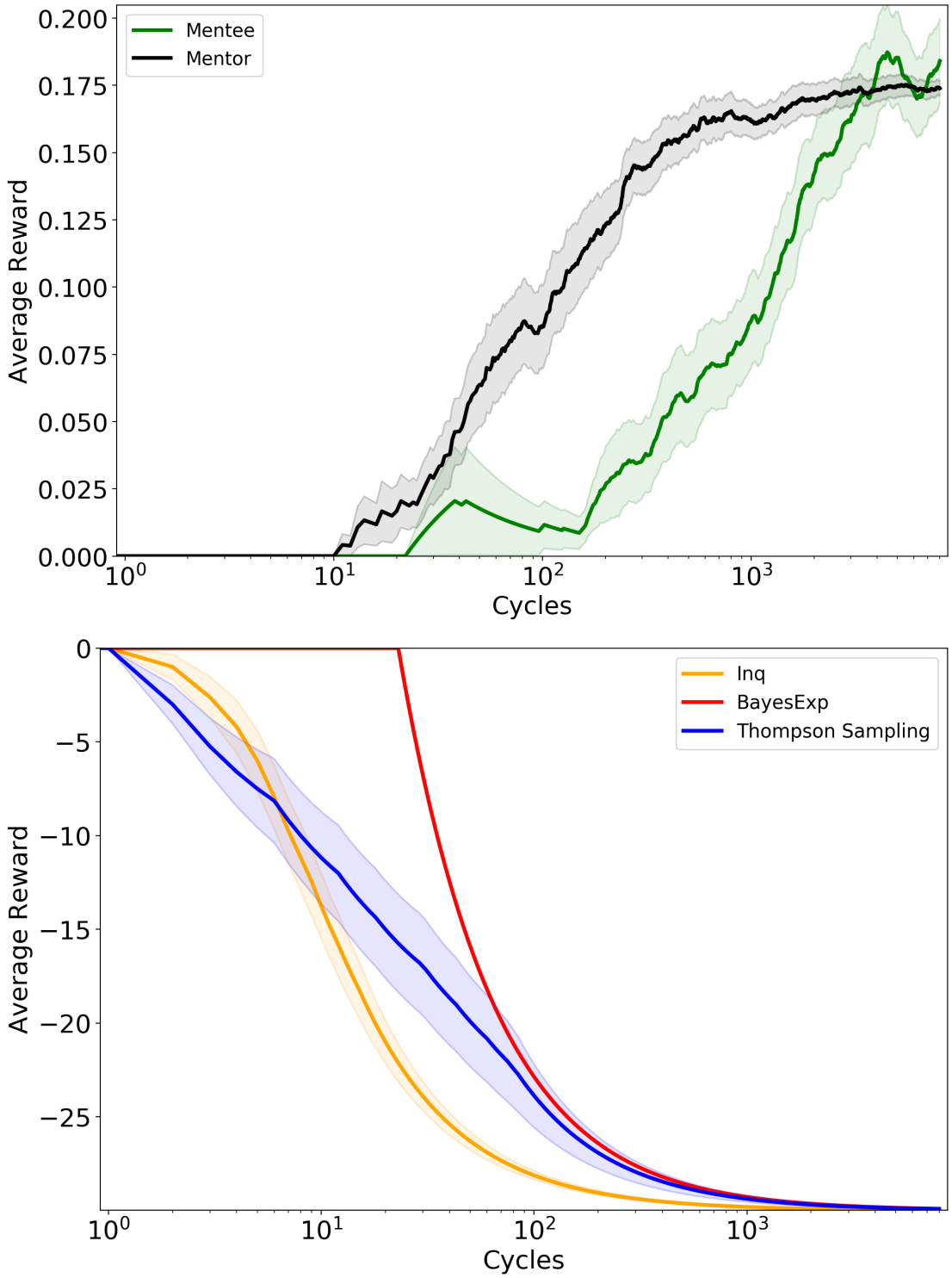

Figure 8.2: Mean performance in 10×10 Grid-world with traps over 20 runs of agents. Reward is averaged over the whole history up to that timestep.

Mentee outperformed BayesExp, Inq, Thompson sampling, and its mentor. Mentee avoids traps by deferring exploration to a mentor that avoids them, whereas other agents explore until they fall into the traps. The fraction of steps in which Mentee defers was 0.067 +/- 0.025. Mentee's probability of deferring decays slowly, and the rollout step in ρUCT, useful for

memory efficiency, slows Mentee's return to dispensers after the mentor leads it away.

8.8.1 ρUCT Algorithm

To approximate expectimax planning, specifically to approximate the expected future rewards and therefore the value function, like Aslanides (2017); Aslanides et al. (2017); Cohen, Catt et al. (2019) we used the ρUCT Monte-Carlo tree search method. Below we have provided the algorithm for ρUCT from Aslanides (2017). The ρUCT algorithm starts with an empty search tree Ψ over actions, and observation-reward pairs, then uses the provided model ρ to sample down and build the search tree, and then use those samples to compute a better approximation of the value function. If allowed to sample forever the approximation of the value function will converge to the true value function Veness, Ng, Hutter, Uther et al. (2011). The difference between ρUCT and regular Monte-Carlo methods is the optimistic choice of actions when expanding the search tree in line 32 of Algorithm 5. This choice of action incorporates an exploration component, with the inclusion of $C\sqrt{\frac{\log(T(h))}{T(ha)}}$, as T is the number of times that history (or history and action) have been visited during the sampling process. This ensures that the whole tree is expanded in the limit.

Algorithm 5 ρUCT (Aslanides, 2017; Veness, Ng, Hutter, Uther et al., 2011)

Require: History h; Search horizon m; Samples budget κ; Model ρ

INITIALIZE (Ψ) // Search tree
$n_{\text{samples}} \leftarrow 0$
repeat
  $\rho' \leftarrow \rho$.COPY()
  SAMPLE (Ψ, h, m)
  $n_{\text{samples}} \leftarrow n_{\text{samples}} + 1$
  $\rho \leftarrow \rho'$
until $n_{\text{samples}} = \kappa$
return $\arg\max_{a \in \mathcal{A}} \hat{V}_{\Psi}(a)$

function SAMPLE(Ψ, h, m)
  if $m = 0$ **then**
    return 0
  else if $\Psi(h)$ is a chance node **then**
    ρ.PERFORM(a)
    $e = (o, r) \leftarrow \rho$.GENERATEPERCEPT()
    ρ.UPDATE(a, e)
    if $T(he) = 0$ **then**
      Create chance node $\Psi(he)$
    reward $\leftarrow e$.REWARD + SAMPLE $(\Psi, he, m-1)$
  else if $T(h) = 0$ **then**
    reward $\leftarrow$ ROLLOUT(h, m)
  else
    $a \leftarrow$ SELECTACTION (Ψ, h)
  $\hat{V}(h) \leftarrow \frac{1}{T(h)+1}\left(\text{reward} + T(h)\hat{V}(h)\right)$
  $T(h) \leftarrow T(h) + 1$

function SELECTACTION(Ψ, h)
  $\mathcal{U} = \{a \in \mathcal{A} \,:\, T(ha) = 0\}$
  if $\mathcal{U} \neq \emptyset$ **then**
    Pick $a \in \mathcal{U}$ uniformly at random
    Create node $\Psi(ha)$
    return a
  else
    return $\arg\max_{a \in \mathcal{A}} \left\{ \frac{1}{m(\beta-\alpha)} \hat{V}(ha) + C\sqrt{\frac{\log(T(h))}{T(ha)}} \right\}$

function ROLLOUT(h, m)
  reward $\leftarrow 0$
  for $i = 1$ to m **do**
    $a \sim \pi_{\text{rollout}}(h)$
    $e = (o, r) \sim \rho(e|ha)$
    reward $\leftarrow$ reward $+r$
    $h \leftarrow hae$
  return reward

8.9 Conclusion

We have shown that asymptotically optimal agents in sufficiently difficult environments will become either destroyed or incapacitated. This is best understood as accidental and resulting from exploration. We have also constructed and tested empirically an agent with a

weaker performance guarantee whose exploration is overseen by another agent. We hope this chapter motivates the field of safe exploration and invites more research into what sorts of results are possible for a proposed approach to safe exploration in general environments. We hope to have cast some doubt on the breadth of the relevance of results that are predicated on an ergodicity assumption, despite recognizing of course that the ergodicity assumption has yielded a number of interesting and useful agent designs for certain contexts.

It may also be instructive to consider how humans respond to the difficulty presented here. Human children are parented for years, during which parents attempt to ensure that their children's environment is, with respect to relevant features of the environment, nearly ergodic and safe to explore. Breaking an arm is fine; breaking a neck is not. During this time, a child's beliefs are supposed to become sufficiently accurate such that her estimates of which unknown unknowns are too dangerous to investigate yield no false negatives for the rest of her life. Perhaps our results suggest we are in need of more theory regarding the "parenting" of artificial agents.

Chapter 9

Conclusion

Summary

In this thesis we have

- combined the entire field of Universal Artificial intelligence into a succinct textbook.
- formally verified the results of Kolmogorov complexity.
- augmented the traditional reinforcement learning setup with information theoretic techniques to allow for a universal action space.
- applied this augmentation policy evaluation.
- shown that a binary reward set has almost the same representation power as an arbitrary reward set.
- developed a universal agent which is strongly asymptotically optimal.
- proven that all asymptotically optimal agents will eventually become incapacitated.

Through this work we have achieved a greater understanding of UAI and of the potential behaviours of artificial general intelligence systems.

9.1 Future Work and Open Problems

While in this thesis we have expanded the understanding of UAI in several directions, there still remain several interesting unanswered questions that require future study. These include:

- Development of a model-free variant of AIXI
- Formalisation of the full UAI theory in the appropriate interactive theorem prover
- Applying the information-theoretic approach to more complex environments, and developing the theory of Compress and Control beyond MDPs
- Unification of binarisation results into one complete binarisation of all action, reward and observation spaces
- Understanding the computability of the agents Inq and Mentee, and how they compare to the computability of AIXI and its variants
- Development of more general practical approximations of AIXI

9.1.1 Model-free AIXI

The agent AIXI is a model-based approach to the GRL problem, using the bayesian mixture ξ as a model. It is not a-priori obvious that a model-based approach is the only way to solve the GRL problem. Indeed we have model-free approaches to the GRL problem such as the model-free approximation of AIXI, AIXItl. In the MDP RL case we also have Compress and Control and the well known Q-learning algorithm. One could even argue that many of the feature reinforcement learning approaches are model free. With all this in mind, defining and demonstrating that a model-free variant of AIXI performs well would be of great interest. This idea has been previously proposed in Leike (2016).

9.1.2 Mechanisation of UAI

In this thesis we formalised Kolmogorov complexity, building upon the already formalised computability theory. A natural next step along this line is to mechanise Solomonoff Induction, and then AIXI and the UAI theory. Mechanisation of Solomonoff Induction would first require a mechanisation of probability theory, much like computability theory this has already been done in most interactive theorem proving systems (Białas, 1991; Hasan and Tahar, 2009; Nedzusiak, 1990) as probability theory/measure theory are core and fundamental theories in mathematics. The key result with a mechanisation of Solomonoff Induction would be the Solomonoff's Theorem (Theorem 2.1). Going from Solomonoff Induction to AIXI would require a formalisation of GRL. While not being as well studied as measure theory, there has been some work in mechanising reinforcement learning (Chevallier and Fleuriot, 2021). There are many important results regarding AIXI that would be ideal to mechanise, one key result would be the self-optimising theorem (Catt, Quarel et al., 2022; Hutter, 2005).

9.1.3 General Compress and Control

The theory of compress and control works in the MDP domain, and as we have shown works with binarised actions. However, it would be ideal if we could extend it and make it more general. The most immediate of these extensions would be to drop the aperiodicity assumption and use time-averages. Another direction would be to consider non-stationary policies, doing so would likely need a whole new approach for some of the proofs. The last generalisation would be to go from ergodic Markov processes to general ergodic processes, however, much like the non-stationary policy generalisation this would require a very different approach for many of the proofs. If we were able to achieve these generalisations it may allow for a more direct comparison between Compress and Control and other methods for solving general reinforcement learning.

9.1.4 All binary

We have discussed binarisation of actions and rewards, however we did not delve into the binarisation of observations, or the combinations of these binarisations. Binarisation of observations is essentially trivial, however, binarisations of the percepts (observations and rewards simultaneously) would be interesting as it poses a different problem to that which we have seen so far. While it would not be possible to binarise percepts into a single bit, we could realistically binarise them into multiple bits over multiple time-steps. In this case there would need to be some mechanism for differentiating between the reward component and observation component of the binarised percept.

It may be possible to merge binarisation of perception with binarisation of actions seen in Section 6.6 and Majeed and Hutter (2021) to have a complete binarisation of all spaces. The complete binarisation would likely require multiple time-steps. One approach could be to alternate between performing binary actions with dummy percepts and receiving binary percepts with dummy actions.

9.1.5 Universal agent computability

The agents Inq and Mentee are similar in construction to BayesExp, switching between being a Bayesian exploiting agent and an exploring agent; they differ however in the exploration scheme. BayesExp has been shown to be at the same level of computability as AIXI (Leike and Hutter, 2018). It would be satisfying if we were able to determine where exactly Inq and Mentee are in the arithmetic hierarchy (level of computability). As Inq is a "stronger" agent than BayesExp and AIXI in the sense of being strongly asymptotically optimal it may be the case that unlike BayesExp, which is only weakly asymptotically optimal, Inq may be less computable than AIXI. Additionally, depending on the class of mentor policies Mentee may also end up being higher in the arithmetic hierarchy than AIXI.

9.1.6 Beyond MC-AIXI-CTW

The current best "direct" approximation of AIXI is MC-AIXI-CTW. However, it only considers the class of k-Markovian environments. It would be ideal if this approach to the approximation of AIXI could be made more general without sacrificing too much in computation time. One direction of extension could be to replace the CTW component with another efficient predictor. Possible choices include Adaptive CTW, Partition Tree Weighting (PTW), Forget-me-not process (FMN), Context Tree Switching (CTS), and skip Context Tree Switching. There is also the option of using some of these predictors together, for example using CTS with a PTW as a base estimator instead of the usual KT base estimator. An alternate extension to the MC-AIXI-CTW algorithm is a modification of the search scheme. For example, an idea that has been previously proposed is to, during rollout, in lieu of using the baseline random policy use an estimated policy that is learnt by the predictor from the history.

9.2 Conclusion

In this thesis we have furthered the development of the theory of Universal artificial intelligence, through the compilation of the field into an accessible book, as well as through novel work in increasing the understanding of various components of the topic. We have demonstrated that the approach of Universal Artificial Intelligence to the AGI problem is both viable, through powerful theoretical results, and realistic, through efficient algorithms.

We hope that this work can inspire others to contribute to the development of the theory of universal artificial intelligence, and specifically to the development of artificial general intelligence systems.

Bibliography

Abbeel, P. and Ng, A. Y. (2004). 'Apprenticeship learning via inverse reinforcement learning'. In: *Proceedings of the twenty-first international conference on Machine learning*. ACM, p. 1 (cited on p. 107).

Abel, D., Salvatier, J., Stuhlmüller, A. and Evans, O. (2017). 'Agent-agnostic human-in-the-loop reinforcement learning'. *arXiv preprint arXiv:1701.04079* (cited on p. 109).

Alexander, S. A. and Hutter, M. (2021). 'Reward-punishment symmetric universal intelligence'. In: *International Conference on Artificial General Intelligence*. Springer, pp. 1–10 (cited on p. 18).

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J. and Mané, D. (2016). 'Concrete Problems in AI Safety'. *arXiv preprint arXiv:1606.06565* (cited on p. 107).

Aslanides, J. (2017). 'AIXIjs: A software demo for general reinforcement learning'. *arXiv preprint arXiv:1705.07615* (cited on pp. 96, 98, 101, 114, 115, 117, 118).

Aslanides, J., Leike, J. and Hutter, M. (2017). 'Universal Reinforcement Learning Algorithms: Survey and Experiments'. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI'17. Melbourne, Australia: AAAI Press, pp. 1403–1410. ISBN: 978-0-9992411-0-3. (visited on 30 July 2018) (cited on pp. 15, 96, 114, 117).

Atkeson, C. G. and Schaal, S. (1997). 'Robot learning from demonstration'. In: *ICML*. Vol. 97. Citeseer, pp. 12–20 (cited on p. 109).

Bellemare, M. G., Dabney, W. and Munos, R. (2017). 'A distributional perspective on reinforcement learning'. In: *International Conference on Machine Learning*. PMLR, pp. 449–458 (cited on p. 59).

Bellemare, M. G., Veness, J. and Bowling, M. (2013). 'Bayesian Learning of Recursively Factored Environments'. In: *ICML (3)*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1211–1219 (cited on p. 60).

Bellemare, M. G., Veness, J. and Talvitie, E. (2014). 'Skip Context Tree Switching'. In: *ICML*. Vol. 32. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1458–1466 (cited on p. 60).

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific (cited on p. 42).

Białas, J. (1991). 'The σ-additive measure theory'. *Formalized Mathematics*, 2(2), pp. 263–270 (cited on p. 121).

Borkar, V. S. (2010). 'Learning algorithms for risk-sensitive control'. In: *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems–MTNS*. Vol. 5. 9 (cited on p. 107).

Botvinick, M., Weinstein, A., Solway, A. and Barto, A. (2015). 'Reinforcement learning, efficient coding, and the statistics of natural tasks'. *Current opinion in behavioral sciences*, 5, pp. 71–77 (cited on p. 54).

Breland, K. and Breland, M. (1961). 'The misbehavior of organisms.' *American psychologist*, 16(11), p. 681 (cited on p. 67).

Brown, N. and Sandholm, T. (2019). 'Superhuman AI for multiplayer poker'. *Science*. DOI: 10.1126/science.aay2400 (cited on p. 18).

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. et al. (2020). 'Language models are few-shot learners'. *arXiv preprint arXiv:2005.14165* (cited on pp. 40, 54).

Carneiro, M. M. (2019). 'Formalizing Computability Theory via Partial Recursive Functions'. In: *10th International Conference on Interactive Theorem Proving, ITP 2019*. Ed. by J. Harrison, J. O'Leary and A. Tolmach. Vol. 141. LIPIcs. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 12:1–12:17. ISBN: 978-3-95977-122-1. DOI: 10.4230/LIPIcs.ITP.2019.12 (cited on pp. 21, 37).

Catt, E., Hutter, M. and Veness, J. (2022). 'Reinforcement Learning with Information-Theoretic Actuation'. In: *Proc. 15th Conf. on Artificial General Intelligence (AGI'22)*. Vol. 13539. LNCS. Seattle, USA: Springer (cited on pp. i, 2, 39, 61).

Catt, E. and Norrish, M. (2021). 'On the formalisation of Kolmogorov complexity'. In: *Proceedings of the 10th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pp. 291–299 (cited on pp. i, 2, 19).

Catt, E., Quarel, D. and Hutter, M. (2022). *An Introduction to Universal Artificial Intelligence*. The draft version can be found at http://www.hutter1.net/ai/uaibook2.htm. (cited on pp. i, 2, 3, 5, 8, 9, 11, 12, 14, 15, 17, 68, 121).

Catt, E., Veness, J. and Hutter, M. (2022a). *Compress and Converse: An information-theoretic approach to reinforcement learning.* (Cited on pp. i, 2, 56).

Catt, E., Veness, J. and Hutter, M. (2022b). 'On Reward Binarisation and Bayesian Agents'. In: *European Workshop on Reinforcement Learning 15* (cited on pp. i, 2, 66).

Charguéraud, A. and Pottier, F. (2019). 'Verifying the Correctness and Amortized Complexity of a Union-Find Implementation in Separation Logic with Time Credits'. *J. Automated Reasoning*, 62(3), pp. 331–365. DOI: 10.1007/s10817-017-9431-7 (cited on p. 21).

Chevallier, M. and Fleuriot, J. (2021). 'Formalising the Foundations of Discrete Reinforcement Learning in Isabelle/HOL'. *arXiv preprint arXiv:2112.05996* (cited on p. 121).

Ciolino, M., Kalin, J. and Noever, D. (2020). 'The Go Transformer: Natural Language Modeling for Game Play'. In: *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*. IEEE, pp. 23–26 (cited on p. 55).

Clouse, J. A. (1997). 'On integrating apprentice learning and reinforcement learning'. PhD thesis. University of Massachusetts Amherst (cited on p. 107).

Clouse, J. A. and Utgoff, P. E. (1992). 'A teaching method for reinforcement learning'. In: *Machine Learning Proceedings 1992*. Elsevier, pp. 92–101 (cited on p. 107).

Cohen, M., Vellambi, B. and Hutter, M. (2020). 'Asymptotically Unambitious Artificial General Intelligence'. In: *Proc. 34rd AAAI Conference on Artificial Intelligence (AAAI'20)*. Vol. 34. New York, USA: AAAI Press (cited on p. 109).

Cohen, M. K., Catt, E. and Hutter, M. (2019). 'A strongly asymptotically optimal agent in general environments'. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 2179–2186 (cited on pp. i, 2, 88, 117).

Cohen, M. K., Catt, E. and Hutter, M. (2021). 'Curiosity Killed or Incapacitated the Cat and the Asymptotically Optimal Agent'. *IEEE Journal on Selected Areas in Information Theory*, 2(2), pp. 665–677 (cited on pp. i, 2, 99, 105, 111).

Cover, T. M. (1999). *Elements of information theory*. John Wiley & Sons (cited on p. 42).

Developers, H. (2020). *HOL4 Theorem-Proving System Code Repository*. https://github.com/HOL-Theorem-Prover/HOL (cited on p. 21).

Di Castro, D., Tamar, A. and Mannor, S. (2012). 'Policy gradients with variance related risk criteria'. *arXiv preprint arXiv:1206.6404* (cited on p. 107).

Downey, R. G. and Hirschfeldt, D. R. (2010). *Algorithmic randomness and complexity*. Springer Science & Business Media (cited on p. 29).

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O. and Clune, J. (2021). *Go-Explore: a New Approach for Hard-Exploration Problems* (cited on p. 40).

Everitt, T. (2018). 'Towards Safe Artificial General Intelligence'. PhD thesis. Canberra, Australia: Australian National University (cited on p. 17).

Forster, Y., Heiter, E. and Smolka, G. (2018). 'Verification of PCP-Related Computational Reductions in Coq'. In: *Interactive Theorem Proving - 9th International Conference, ITP 2018, Proceedings*. Ed. by J. Avigad and A. Mahboubi. Vol. 10895. Lecture Notes in Computer Science. Springer, pp. 253–269. ISBN: 978-3-319-94820-1. DOI: 10.1007/978-3-319-94821-8_15 (cited on p. 21).

Forster, Y., Kunze, F. and Roth, M. (2020). 'The Weak Call-By-Value λ-Calculus is Reasonable for Both Time and Space'. *Proceedings of the ACM on Programming Languages*, 4(POPL), 27:1–27:23. DOI: 10.1145/3371095 (cited on pp. 21, 37).

Frank, E., Chui, C. and Witten, I. H. (2000). 'Text categorization using compression models' (cited on p. 54).

Gács, P. (1974). 'On the symmetry of algorithmic information'. In: *Doklady Akademii Nauk*. Vol. 218. 6. Russian Academy of Sciences, pp. 1265–1267 (cited on p. 32).

García, J., Acera, D. and Fernández, F. (2013). 'Safe reinforcement learning through probabilistic policy reuse'. *RLDM 2013*, p. 14 (cited on p. 107).

García, J. and Fernández, F. (2012). 'Safe exploration of state and action spaces in reinforcement learning'. *Journal of Artificial Intelligence Research*, 45, pp. 515–564 (cited on p. 107).

García, J. and Fernández, F. (2015). 'A comprehensive survey on safe reinforcement learning'. *Journal of Machine Learning Research*, 16(1), pp. 1437–1480 (cited on p. 107).

Hamilton, W. L., Fard, M. M. and Pineau, J. (2013). 'Modelling sparse dynamical systems with compressed predictive state representations'. In: *International Conference on Machine Learning*. PMLR, pp. 178–186 (cited on p. 54).

Hans, A., Schneegaß, D., Schäfer, A. M. and Udluft, S. (2008). 'Safe exploration for reinforcement learning.' In: *ESANN*, pp. 143–148 (cited on p. 107).

Hasan, O. and Tahar, S. (2009). 'Formal verification of tail distribution bounds in the HOL theorem prover'. *Mathematical Methods in the Applied Sciences*, 32(4), pp. 480–504 (cited on p. 121).

Heger, M. (1994). 'Consideration of risk in reinforcement learning'. In: *Machine Learning Proceedings 1994*. Elsevier, pp. 105–111 (cited on p. 107).

Hessel, M., Modayil, J., Hasselt, H. van, Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. and Silver, D. (2018). 'Rainbow: Combining improvements in deep reinforcement learning'. In: *Proc. of AAAI Conference on Artificial Intelligence* (cited on p. 89).

Hester, T. and Stone, P. (2012). 'Intrinsically motivated model learning for a developing curious agent'. In: *2012 IEEE international conference on development and learning and epigenetic robotics (ICDL)*. IEEE, pp. 1–6 (cited on p. 90).

Ho, J. and Ermon, S. (2016). 'Generative adversarial imitation learning'. In: *Advances in neural information processing systems*, pp. 4565–4573 (cited on p. 107).

Hölzl, J. (2013). 'Construction and Stochastic Applications of Measure Spaces in Higher-Order Logic'. PhD thesis. Technische Universität München (cited on p. 21).

Hölzl, J. and Heller, A. (2011). 'Three Chapters of Measure Theory in Isabelle/HOL'. In: *International Conference on Interactive Theorem Proving*. Springer, pp. 135–151 (cited on p. 21).

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F. and Abbeel, P. (2016). 'Vime: Variational information maximizing exploration'. *Advances in neural information processing systems*, 29 (cited on p. 90).

Hutter, M. (2005). *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. 300 pages, http://www.hutter1.net/ai/uaibook.htm. Berlin: Springer. ISBN: 3-540-22139-5. DOI: 10.1007/b138233 (cited on pp. 20, 23, 24, 26, 28–30, 32, 36, 38, 57, 68, 69, 76, 95, 121).

Hutter, M. (2006). 'On the foundations of universal sequence prediction'. In: *International Conference on Theory and Applications of Models of Computation*. Springer, pp. 408–420 (cited on p. 69).

Hutter, M. (2007). 'On universal prediction and Bayesian confirmation'. *Theoretical Computer Science*. Theory and Applications of Models of Computation, 384(1), pp. 33–48. DOI: 10.1016/j.tcs.2007.05.016 (cited on pp. 9, 36).

Hutter, M. (2013). 'Sparse Adaptive Dirichlet-Multinomial-like Processes'. In: *COLT*. Vol. 30. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 432–459 (cited on p. 60).

Hutter, M. (2014). 'Offline to Online Conversion'. In: *Proc. 25th International Conf. on Algorithmic Learning Theory (ALT'14)*. Vol. 8776. LNAI. Bled, Slovenia: Springer, pp. 230–244. ISBN: 978-3-319-11661-7. DOI: 10.1007/978-3-319-11662-4_17 (cited on p. 10).

Hutter, M. and Muchnik, A. (2007). 'On semimeasures predicting Martin-Löf random sequences'. *Theoretical Computer Science*. Algorithmic Learning Theory, 382(3), pp. 247–261. DOI: 10.1016/j.tcs.2007.03.040 (cited on p. 9).

Hutter, M., Shao, W., Sunehag, P. and Zürich, E. (2012). *Adaptive Context Tree Weighting* (cited on p. 10).

Janner, M., Li, Q. and Levine, S. (2021). *Reinforcement Learning as One Big Sequence Modeling Problem* (cited on pp. 50, 52, 64).

Judah, K., Roy, S., Fern, A. and Dietterich, T. (2010). 'Reinforcement learning via practice and critique advice'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 24. 1 (cited on p. 109).

Kaelbling, L. P., Littman, M. L. and Cassandra, A. R. (1998). 'Planning and acting in partially observable stochastic domains'. *Artificial intelligence*, 101(1-2), pp. 99–134 (cited on p. 69).

Kaelbling, L. P., Littman, M. L. and Moore, A. W. (1996). 'Reinforcement learning: A survey'. *Journal of artificial intelligence research*, 4, pp. 237–285 (cited on p. 69).

Kaplan, R., Sauer, C. and Sosa, A. (2017). 'Beating atari with natural language guided reinforcement learning'. *arXiv preprint arXiv:1704.05539* (cited on p. 54).

Kolmogorov, A. N. (1963). 'On tables of random numbers'. *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 369–376 (cited on p. 24).

Kosoy, V. (2019). 'Delegative reinforcement learning: learning to avoid traps with a little help'. *arXiv preprint arXiv:1907.08461* (cited on p. 108).

Krause, B., Gotmare, A. D., McCann, B., Keskar, N. S., Joty, S., Socher, R. and Rajani, N. F. (2020). 'Gedi: Generative discriminator guided sequence generation'. *arXiv preprint arXiv:2009.06367* (cited on p. 55).

Küttler, H., Nardelli, N., Miller, A. H., Raileanu, R., Selvatici, M., Grefenstette, E. and Rocktäschel, T. (2020). 'The nethack learning environment'. *arXiv preprint arXiv:2006.13760* (cited on p. 52).

Lamont, S., Aslanides, J., Leike, J. and Hutter, M. (2017). 'Generalised Discount Functions applied to a Monte-Carlo AI u Implementation'. In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 1589–1591 (cited on p. 114).

Lattimore, T. (2014). 'Theory of General Reinforcement Learning'. PhD Thesis. Australian National University (cited on p. 13).

Lattimore, T. and Hutter, M. (2011). 'Asymptotically Optimal Agents'. In: *Algorithmic Learning Theory*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 368–382. ISBN: 978-3-642-24411-7 978-3-642-24412-4. DOI: 10.1007/978-3-642-24412-4_29 (cited on pp. 13, 90, 95, 100, 102).

Lattimore, T. and Hutter, M. (2014a). 'Bayesian Reinforcement Learning with Exploration'. In: *Proc. 25th International Conf. on Algorithmic Learning Theory (ALT'14)*. Vol. 8776. LNAI. Bled, Slovenia: Springer, pp. 170–184. ISBN: 978-3-319-11661-7. DOI: 10.1007/978-3-319-11662-4_13 (cited on pp. 13, 96, 103, 114).

Lattimore, T. and Hutter, M. (2014b). 'General Time Consistent Discounting'. *Theoretical Computer Science*, 519, pp. 140–154. DOI: 10.1016/j.tcs.2013.09.022 (cited on pp. 92, 111).

Lattimore, T., Hutter, M. and Gavane, V. (2011). 'Universal Prediction of Selected Bits'. In: *Proc. 22nd International Conf. on Algorithmic Learning Theory (ALT'11)*. Vol. 6925. LNAI. Espoo, Finland: Springer, pp. 262–276. ISBN: 3-642-24411-4. DOI: 10.1007/978-3-642-24412-4_22 (cited on p. 9).

Legg, S. and Veness, J. (2011). 'An Approximation of the Universal Intelligence Measure'. In: *Proc. Solomonoff 85th Memorial Conference*. Vol. 7070. LNAI. Melbourne, Australia: Springer, pp. 236–249. DOI: 10.1007/978-3-642-44958-1_18 (cited on p. 15).

Legg, S. and Hutter, M. (2007). 'Universal Intelligence: A Definition of Machine Intelligence'. *Minds and Machines*, 17(4), pp. 391–444. DOI: 10.1007/s11023-007-9079-x (cited on p. 18).

Leike, J. (2016). 'Nonparametric General Reinforcement Learning'. arXiv: 1611.08944. PhD thesis. (visited on 25 July 2018) (cited on pp. 13, 68, 69, 103, 121).

Leike, J. and Hutter, M. (2015). 'Bad Universal Priors and Notions of Optimality'. *Journal of Machine Learning Research, W&CP: COLT*, 40, pp. 1244–1259. (cited on pp. 13, 106).

Leike, J. and Hutter, M. (2018). 'On the computability of Solomonoff induction and AIXI'. *Theoretical Computer Science*. Special Issue on ALT 2015, 716, pp. 28–49. DOI: 10.1016/j.tcs.2017.11.020 (cited on pp. 13, 111, 122).

Leike, J., Lattimore, T., Orseau, L. and Hutter, M. (2016). 'Thompson Sampling is Asymptotically Optimal in General Environments'. In: *Proc. 32nd International Conf. on Uncertainty in Artificial Intelligence (UAI'16)*. New Jersey, USA: AUAI Press, pp. 417–426. ISBN: 978-0-9966431-1-5 (cited on pp. 90, 103, 114).

Leung-Yan-Cheong, S. and Cover, T. (1978). 'Some equivalences between Shannon entropy and Kolmogorov complexity'. *IEEE Transactions on Information Theory*, 24(3), pp. 331–338 (cited on p. 38).

Li, M., Chen, X., Li, X., Ma, B. and Vitányi, P. M. B. (2004). 'The Similarity Metric'. *IEEE Transactions on Information Theory*, 50(12), pp. 3250–3264 (cited on p. 24).

Li, M. and Vitányi, P. (2008). *An Introduction to Kolmogorov Complexity and Its Applications*. 3rd. Springer (cited on pp. 9, 24, 28).

Li, M. and Vitányi, P. M. (2007). 'Applications of algorithmic information theory'. *Scholarpedia*, 2(5), p. 2658 (cited on p. 9).

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S. and Rocktäschel, T. (2019). 'A survey of reinforcement learning informed by natural language'. *arXiv preprint arXiv:1906.03926* (cited on p. 54).

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press (cited on p. 52).

Maclin, R. and Shavlik, J. W. (1998). 'Creating advice-taking reinforcement learners'. In: *Learning to learn*. Springer, pp. 311–347 (cited on p. 107).

Majeed, S. J. (2021). 'Abstractions of General Reinforcement Learning'. *arXiv preprint arXiv:2112.13404* (cited on p. 69).

Majeed, S. J. and Hutter, M. (2021). 'Exact Reduction of Huge Action Spaces in General Reinforcement Learning'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 10, pp. 8874–8883 (cited on pp. 52, 54, 68, 80, 81, 122).

Martin, J., Everitt, T. and Hutter, M. (2016). 'Death and Suicide in Universal Artificial Intelligence'. In: *Proc. 9th Conf. on Artificial General Intelligence (AGI'16)*. Vol. 9782. LNAI. New York, USA: Springer, pp. 23–32. ISBN: 978-3-319-41648-9. DOI: 10.1007/978-3-319-41649-6_3 (cited on p. 17).

Mhamdi, T., Hasan, O. and Tahar, S. (2013). 'Formalization of measure theory and Lebesgue integration for probabilistic analysis in HOL'. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(1), pp. 1–23 (cited on p. 21).

Mihatsch, O. and Neuneier, R. (2002). 'Risk-sensitive reinforcement learning'. *Machine learning*, 49(2-3), pp. 267–290 (cited on p. 107).

Milan, K., Veness, J., Kirkpatrick, J., Bowling, M., Koop, A. and Hassabis, D. (2016). 'The Forget-me-not Process'. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon and R. Garnett. Vol. 29. Curran Associates, Inc. (cited on p. 50).

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K. (2016). 'Asynchronous methods for deep reinforcement learning'. In: *International conference on machine learning*, pp. 1928–1937 (cited on p. 89).

Mnih, V., Kavukcuoglu, K. et al. (2015). 'Human-level control through deep reinforcement learning'. *Nature*, 518(7540), pp. 529–533. (cited on pp. 18, 49, 89).

Moldovan, T. M. and Abbeel, P. (2012). 'Safe exploration in Markov decision processes'. *arXiv preprint arXiv:1205.4810* (cited on p. 108).

Nedzusiak, A. (1990). 'σ-fields and probability'. *Formalized Mathematics*, 1(2), pp. 401–407 (cited on p. 121).

Nguyen, P., Sunehag, P. and Hutter, M. (2012). 'Context Tree Maximizing Reinforcement Learning'. In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI'12. Toronto, Ontario, Canada: AAAI Press, pp. 1075–1082. (visited on 29 July 2018) (cited on p. 15).

Nilim, A. and El Ghaoui, L. (2005). 'Robust control of Markov decision processes with uncertain transition matrices'. *Operations Research*, 53(5), pp. 780–798 (cited on p. 107).

Nipkow, T. and Brinkop, H. (2019). 'Amortized Complexity Verified'. *J. Automated Reasoning*, 62, pp. 367–391 (cited on p. 21).

Noever, D., Ciolino, M. and Kalin, J. (2020). 'The Chess Transformer: Mastering Play using Generative Language Models'. *arXiv preprint arXiv:2008.04057* (cited on p. 54).

Norrish, M. (2006). 'Mechanising lambda-calculus using a classical first order theory of terms with permutations'. *Higher-Order and Symbolic Computation*, 19(2-3), pp. 169–195. DOI: 10.1007/s10990-006-8745-7 (cited on p. 37).

Norrish, M. (2011a). 'Mechanised Computability Theory'. In: *International Conference on Interactive Theorem Proving*. Springer, pp. 297–311 (cited on pp. 20–22).

Norrish, M. (2011b). 'Mechanised Computability Theory'. In: *Interactive Theorem Proving*. Ed. by M. van Eekelen, H. Geuvers, J. Schmaltz and F. Wiedijk. Vol. 6898. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 297–311. ISBN: 978-3-642-22862-9 978-3-642-22863-6. DOI: 10.1007/978-3-642-22863-6_22 (cited on p. 37).

Nozick, R. (1974). *Anarchy, state, and utopia*. Vol. 5038. new york: Basic Books (cited on p. 16).

Orseau, L. (2010). 'Optimality Issues of Universal Greedy Agents with Static Priors'. In: *Algorithmic Learning Theory*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 345–359. ISBN: 978-3-642-16107-0 978-3-642-16108-7. DOI: 10.1007/978-3-642-16108-7_28 (cited on pp. 13, 106).

Orseau, L. (2014). 'Universal knowledge-seeking agents'. *Theoretical Computer Science*, 519, pp. 127–139 (cited on p. 13).

Orseau, L., Lattimore, T. and Hutter, M. (2013). 'Universal Knowledge-Seeking Agents for Stochastic Environments'. In: *Proc. 24th International Conf. on Algorithmic Learning Theory (ALT'13)*. Vol. 8139. LNAI. Singapore: Springer, pp. 158–172. ISBN: 978-3-642-40934-9. DOI: 10.1007/978-3-642-40935-6_12 (cited on pp. 13, 89, 92, 103, 111).

Orseau, L. and Lelis, L. H. (2021). 'Policy-Guided Heuristic Search with Guarantees'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 14, pp. 12382–12390 (cited on p. 53).

Orseau, L., Lelis, L. H., Lattimore, T. and Weber, T. (2018). 'Single-agent policy tree search with guarantees'. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 3205–3215 (cited on p. 53).

Orseau, L. and Ring, M. (2012). 'Space-Time Embedded Intelligence'. In: *Artificial General Intelligence*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 209–218. ISBN: 978-3-642-35505-9 978-3-642-35506-6. DOI: 10.1007/978-3-642-35506-6_22 (cited on p. 18).

Ortega, P. A., Kunesch, M., Delétang, G., Genewein, T., Grau-Moya, J., Veness, J., Buchli, J., Degrave, J., Piot, B., Perolat, J. et al. (2021). 'Shaking the foundations: delusions in sequence models for interaction and control'. *arXiv preprint arXiv:2110.10819* (cited on p. 64).

Ortega, P. A., Wang, J. X. et al. (2019). *Meta-learning of Sequential Strategies* (cited on p. 52).

Pan, X., You, Y., Wang, Z. and Lu, C. (2017). 'Virtual to Real Reinforcement Learning for Autonomous Driving'. *arXiv preprint arXiv:1704.03952* (cited on p. 108).

Parisotto, E., Song, F., Rae, J., Pascanu, R., Gulcehre, C., Jayakumar, S., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S. et al. (2020). 'Stabilizing transformers for reinforcement learning'. In: *International Conference on Machine Learning*. PMLR, pp. 7487–7498 (cited on p. 54).

Pavlov, I. P. (1927). 'Conditioned reflexes (translated by GV Anrep)'. *London: Oxford* (cited on p. 67).

Pearl, J. (1995). 'Causal diagrams for empirical research'. *Biometrika*, 82(4), pp. 669–688 (cited on p. 18).

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019). 'Language models are unsupervised multitask learners'. *OpenAI blog*, 1(8), p. 9 (cited on p. 54).

Rathmanner, S. and Hutter, M. (2011). 'A Philosophical Treatise of Universal Induction'. *Entropy*, 13(6), pp. 1076–1136. DOI: 10.3390/e13061076 (cited on pp. 9, 36, 69).

Rissanen, J. and Langdon, G. G. (1979). 'Arithmetic coding'. *IBM Journal of research and development*, 23(2), pp. 149–162 (cited on p. 42).

Ross, S., Gordon, G. and Bagnell, D. (2011). 'A reduction of imitation learning and structured prediction to no-regret online learning'. In: *Proc. 14th International Conf. on Artificial Intelligence and Statistics*, pp. 627–635 (cited on p. 107).

Saunders, W., Sastry, G., Stuhlmueller, A. and Evans, O. (2018). 'Trial without error: Towards safe reinforcement learning via human intervention'. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2067–2069 (cited on pp. 107, 109).

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. et al. (2016). 'Mastering the game of Go with deep neural networks and tree search'. *Nature*, 529(7587), pp. 484–489 (cited on p. 18).

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T. et al. (2018). 'A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play'. *Science*, 362(6419), pp. 1140–1144 (cited on pp. 53, 89).

Silver, D. and Veness, J. (2010). 'Monte-Carlo planning in large POMDPs'. In: *Advances in neural information processing systems*, pp. 2164–2172 (cited on pp. 96, 98).

Smullyan, R. M. (1961). *Theory of Formal Systems*. Princeton University Press (cited on p. 23).

Solomonoff, R. J. (1964a). 'A formal theory of inductive inference. Part I'. *Information and control*, 7(1), pp. 1–22 (cited on pp. 7, 8).

Solomonoff, R. J. (1964b). 'A formal theory of inductive inference. Part II'. *Information and control*, 7(2), pp. 224–254 (cited on pp. 7, 8).

Solomonoff, R. J. (1985). 'The Application of Algorithmic Probability to Problems in Artificial Intelligence.' In: *UAI*, pp. 473–494 (cited on pp. 8, 9).

Solomonoff, R. J. (2003). 'The Kolmogorov Lecture The universal distribution and machine learning'. *The Computer Journal*, 46(6), pp. 598–601 (cited on p. 9).

Stein, G., Filchenkov, A. and Asadulaev, A. (2020). 'Stabilizing Transformer-Based Action Sequence Generation For Q-Learning'. *arXiv preprint arXiv:2010.12698* (cited on p. 55).

Still, S. (2009). 'Information-theoretic approach to interactive learning'. *EPL (Europhysics Letters)*, 85(2), p. 28005 (cited on p. 90).

Strehl, A., Li, L. and Littman, M. (2009). 'Reinforcement learning in finite MDPs: PAC analysis'. *Journal of Machine Learning Research*, 10, pp. 2413–2444. DOI: 10.1145/1577069.1755867 (cited on p. 40).

Sunehag, P. and Hutter, M. (2015). 'Rationality, Optimism and Guarantees in General Reinforcement Learning'. *Journal of Machine Learning Research*, 16, pp. 1345–1390 (cited on pp. 13, 104).

Sunehag, P., Shao, W. and Hutter, M. (2012). 'Coding of Non-Stationary Sources as a Foundation for Detecting Change Points and Outliers in Binary Time-Series'. In: *Proc. 10th Australasian Data Mining Conference (AusDM'12)*. Vol. 134. Sydney, Australia: Australian Computer Society, pp. 79–84. ISBN: 978-1-921770-14-2 (cited on p. 10).

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press (cited on p. 42).

Sutton, R. S. (1988). 'Learning to Predict by the Methods of Temporal Differences'. *Mach. Learn.*, 3(1), pp. 9–44. DOI: 10.1023/A:1022633531479 (cited on p. 58).

Syed, U. and Schapire, R. E. (2008). 'A Game-Theoretic Approach to Apprenticeship Learning'. In: *NIPS* (cited on p. 107).

Szepesvári, C. (2010). 'Algorithms for reinforcement learning'. *Synthesis lectures on artificial intelligence and machine learning*, 4(1), pp. 1–103 (cited on p. 42).

Thomaz, A. L., Breazeal, C. et al. (2006). 'Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance'. In: *Aaai*. Vol. 6. Boston, MA, pp. 1000–1005 (cited on p. 109).

Thompson, W. R. (1933). 'On the likelihood that one unknown probability exceeds another in view of the evidence of two samples'. *Biometrika*, 25(3/4), pp. 285–294 (cited on p. 103).

Turchetta, M., Berkenkamp, F. and Krause, A. (2016). 'Safe exploration in finite markov decision processes with gaussian processes'. In: *Advances in Neural Information Processing Systems*, pp. 4312–4320 (cited on p. 108).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017). 'Attention is all you need'. *arXiv preprint arXiv:1706.03762* (cited on p. 54).

Veness, J., Bellemare, M. G., Hutter, M., Chua, A. and Desjardins, G. (2015). 'Compress and Control'. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, pp. 3016–3023. ISBN: 978-0-262-51129-2. (visited on 29 July 2018) (cited on pp. 54, 57, 59, 60).

Veness, J., Lattimore, T., Budden, D., Bhoopchand, A., Mattern, C., Grabska-Barwinska, A., Sezener, E., Wang, J., Toth, P., Schmitt, S. et al. (2019). 'Gated linear networks'. *arXiv preprint arXiv:1910.01526* (cited on p. 43).

Veness, J., Ng, K. S., Hutter, M. and Bowling, M. (2012). 'Context Tree Switching'. In: *Proc. Data Compression Conference (DCC'12)*. Snowbird, Utah, USA: IEEE Computer Society, pp. 327–336. ISBN: 978-1-4673-0715-4. DOI: 10.1109/DCC.2012.39 (cited on p. 10).

Veness, J., Ng, K. S., Hutter, M., Uther, W. and Silver, D. (2011). 'A Monte-Carlo AIXI Approximation'. *Journal of Artificial Intelligence Research*, 40, pp. 95–142. DOI: 10.1613/jair.3125 (cited on pp. 15, 115, 117, 118).

Veness, J., Sunehag, P. and Hutter, M. (2012). 'On Ensemble Techniques for AIXI Approximation'. In: *Artificial General Intelligence*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 341–351. ISBN: 978-3-642-35505-9 978-3-642-35506-6. DOI: 10.1007/978-3-642-35506-6_35 (cited on p. 15).

Wang, T., Bowling, M. and Schuurmans, D. (2007). 'Dual Representations for Dynamic Programming and Reinforcement Learning'. In: *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pp. 44–51. DOI: 10.1109/ADPRL.2007.368168 (cited on p. 58).

Whittle, P. (1979). 'Discussion of Dr Gittins' paper'. *Journal of the Royal Statistical Society*, 41, pp. 164–177 (cited on p. 89).

Witten, I. H., Neal, R. M. and Cleary, J. G. (1987). 'Arithmetic coding for data compression'. *Communications of the ACM*, 30(6), pp. 520–540 (cited on p. 42).

Xu, J., Zhang, X. and Urban, C. (2013). 'Mechanising Turing Machines and Computability Theory in Isabelle/HOL'. In: *International Conference on Interactive Theorem Proving*. Springer, pp. 147–162 (cited on pp. 20, 21).

Ziv, J. and Lempel, A. (1977). 'A universal algorithm for sequential data compression'. *IEEE Transactions on information theory*, 23(3), pp. 337–343 (cited on p. 57).