Doctoral thesis

Towards Safe Artificial General Intelligence

Tom Everitt

Submitted May, 2018
minor revision June, 2019

A thesis submitted for the degree of
Doctor of Philosophy
at the
Australian National University

Several of the chapters of this thesis are based on joint work with other students and researchers. Except for Chapter 11, I was responsible for proving the main results and writing most of the papers. The paper behind Chapter 11 is based on my idea, and I have heavily revised the text before putting it into this thesis. I am indebted to all my co-authors for contributing their great ideas and suggestions, as well as for making the work much more enjoyable. To the best of my knowledge the results in this thesis are my own except where a reference is provided.

Tom Everitt, May 2018

Acknowledgments

I wish to thank the following people for in one way or another contributing to my writing this thesis:

My supervisor Prof. Marcus Hutter, for letting me try my luck at a topic that was very "fringe" back in 2015 when I started, and who pushed, guided, criticized, and supported me throughout the journey. Now, whenever I write an argument, I ask myself: "Would Marcus be convinced by this?" Often the answer is no. I believe this makes my writing a lot better.

My other panel members Laurent Orseau and Stephen Gould. Laurent's deep knowledge about AGI safety has been especially valuable.

My fellow PhD students Jan Leike and Mayank Daswani for persuading me that AGI safety was an important and worthwhile PhD topic. It took a while, but eventually I got the point.

The DeepMind AGI safety team for hosting me for an internship, and MIRI and CFAR for a colloqium series and a summer fellowship. All were very inspiring, and helped me feel part of a larger AGI safety community.

Björn Wängberg and Martin Asperholm for many hours of fierce but highly enjoyable conversations that forged my intellectual foundations.

The Canberra Underdogs underwater rugby team and ANU Lunchtime Soccer for being great counterbalances to intellectual work.

My wife and daughter, Tianwei and Maia, for picking up the pieces of my overworked brain after every paper deadline. Thanks for being a solid base for recovery, and for enduring my absent mind with unwavering spirit.

My family in Sweden, for making me who I am.

Though not forgotten, many other teachers, collaborators, friends, and organizations must go unnamed. In particular, millions of Australian tax payers generously funded my PhD education,[1] and millions of Swedish ones funded my earlier education. If it takes a village to raise a child, then it takes a civilization to raise a researcher.

[1] I received the ANU University Research Scholarship (International) (669/2014) and an HDR Fee Remission Merit Scholarship (2712014).

Abstract

The field of artificial intelligence has recently experienced a number of breakthroughs thanks to progress in deep learning and reinforcement learning. Computer algorithms now outperform humans at Go, Jeopardy, image classification, and lip reading, and are becoming very competent at driving cars and interpreting natural language. The rapid development has led many to conjecture that artificial intelligence with greater-than-human ability on a wide range of tasks may not be far. This in turn raises concerns whether we know how to control such systems, in case we were to successfully build them.

Indeed, if humanity would find itself in conflict with a system of much greater intelligence than itself, then human society would likely lose. One way to make sure we avoid such a conflict is to ensure that any future AI system with potentially greater-than-human-intelligence has goals that are *aligned* with the goals of the rest of humanity. For example, it should not wish to kill humans or steal their resources.

The main focus of this thesis will therefore be *goal alignment*, i.e. how to design artificially intelligent agents with goals coinciding with the goals of their designers. Focus will mainly be directed towards variants of reinforcement learning, as reinforcement learning currently seems to be the most promising path towards powerful artificial intelligence. We identify and categorize goal misalignment problems in reinforcement learning agents as designed today, and give examples of how these agents may cause catastrophes in the future. We also suggest a number of reasonably modest modifications that can be used to avoid or mitigate each identified misalignment problem. Finally, we also study various choices of decision algorithms, and conditions for when a powerful reinforcement learning system will permit us to shut it down.

The central conclusion is that while reinforcement learning systems as designed today are inherently unsafe to scale to human levels of intelligence, there are ways to potentially address many of these issues without straying too far from the currently so successful reinforcement learning paradigm. Much work remains in turning the high-level proposals suggested in this thesis into practical algorithms, however.

Central claim: *There are a number of theoretically valid, partial solutions to the problem of keeping artificial general intelligence both safe and useful.*

Contents

List of Tables

List of Figures

List of Publications

This thesis is based on the papers:

- Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg (2017). "Reinforcement Learning with Corrupted Reward Signal". In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 4705–4713. arXiv: 1705.08417

- Tom Everitt, Gary Lea, and Marcus Hutter (2018). "AGI Safety Literature Review". In: *International Joint Conference on Artificial Intelligence (IJCAI)*

- Tom Everitt, Daniel Filan, Mayank Daswani, and Marcus Hutter (2016). "Self-modification of policy and utility function in rational agents". In: *Artificial General Intelligence*. Vol. LNAI 9782, pp. 1–11. ISBN: 9783319416489. arXiv: 1605.03142. *Winner of the Kurzweil Prize for best AGI Paper.*

- Tom Everitt and Marcus Hutter (submitted 2018). "The Alignment Problem for Bayesian History-Based Reinforcement Learners". URL: https://www.tomeveritt.se/papers/alignment.pdf. *Winner of the AI Alignment Prize.*[1]

- Tom Everitt and Marcus Hutter (2018). "Universal Artificial Intelligence: Practical Agents and Fundamental Challengs". In: *Foundations of Trusted Autonomy*. Ed. by Hussein A. Abbass, Jason Scholz, and Darryn J. Reid. Springer. Chap. 2, pp. 15–46. ISBN: 978-3-319-64816-3

- Tom Everitt, Jan Leike, and Marcus Hutter (2015). "Sequential Extensions of Causal and Evidential Decision Theory". In: *Algorithmic Decision Theory*. Ed. by Toby Walsh. Springer, pp. 205–221. arXiv: 1506.07359

- Tom Everitt and Marcus Hutter (2016). "Avoiding wireheading with value reinforcement learning". In: *Artificial General Intelligence*. Vol. LNAI 9782, pp. 12–22. ISBN: 9783319416489. arXiv: 1605.03143

[1] Will be published as Tom Everitt and Marcus Hutter (forthcoming). *Reward Tampering Problems and Solutions in Reinforcement Learning: A Causal Influence Digaram Perspective.*

- Tobias Wängberg, Mikael Böörs, Elliot Catt, Tom Everitt, and Marcus Hutter (2017). "A Game-Theoretic Analysis of the Off-Switch Game". In: *Artificial General Intelligence*. Springer, pp. 167–177. arXiv: 1708.03871

During my PhD I have also published:

- Tom Everitt, Ben Goertzel, and Alexey Potapov, eds. (2017). *Artificial General Intelligence: 10th International Conference, AGI 2017, Melbourne, VIC, Australia, August 15-18, 2017, Proceedings*. Springer. ISBN: 978-3-319-63702-0

- Tom Everitt and Marcus Hutter (2015b). "Analytical Results on the BFS vs. DFS Algorithm Selection Problem. Part I: Tree Search". In: *28th Australasian Joint Conference on Artificial Intelligence*, pp. 157–165

- Tom Everitt and Marcus Hutter (2015c). "Analytical Results on the BFS vs. DFS Algorithm Selection Problem. Part II: Graph Search". In: *28th Australasian Joint Conference on Artificial Intelligence*, pp. 166–178

- Tom Everitt and Marcus Hutter (2015a). *A Topological Approach to Meta-heuristics: Analytical Results on the BFS vs. DFS Algorithm Selection Problem*. Tech. rep. Australian National University, pp. 1–36. arXiv: 1509.02709

- Jarryd Martin, Tom Everitt, and Marcus Hutter (2016). "Death and Suicide in Universal Artificial Intelligence". In: *Artificial General Intelligence*. Springer, pp. 23–32. arXiv: 1606.00652

- Jarryd Martin, Suraj Sasikumar, Tom Everitt, and Marcus Hutter (2017). "Count-Based Exploration in Feature Space for Reinforcement Learning". In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 2471–2478. arXiv: 1706.08090

- Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A. Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg (2017). *AI Safety Gridworlds*. arXiv: 1711.09883

Part I.

Background

To educate a machine in mind and not in morals is to educate a menace to society.

Theodore Roosevelt (paraphrased)

1. Introduction[1]

Artificial Intelligence. An artificial intelligence (AI) program may broadly be defined as a computer program that automatically finds ways to achieve goals. For example, an AI chess program finds moves for achieving the goal of checkmating the opponent. The moves have not been explicitly defined by the programmer – instead the program finds them by "weighing" the pros and cons of different strategies. An AI question-answering system automatically generates the answer to a query by searching a knowledge base. A self-driving car finds a way to control throttle, break, and steering, in order to get to its goal destination.

An important distinction between different AI systems is generality. Most present-day AIs can only solve narrow sets of tasks in restricted environments. For example, most chess programs can only play chess, and most question answering systems can only answer specific type of queries from a particular kind of database. This is in contrast to humans, who can learn to perform a wide range of cognitive tasks.

However, a trend towards greater generality can be observed in AI systems. While most self-driving car algorithms would not perform well on tasks other than driving a car, driving a car is already a rather general task. It involves acting in an environment with varying weather, traffic, and lightning conditions, using multiple sources of perception. While most board-game AIs have traditionally been tailored towards a specific type of board game, AlphaZero is a single algorithm that achieves super human performance in Go, Chess, and Shogi (Silver, Hubert, et al., 2017). Another popular benchmark is video games. Here again a single algorithm called DQN recently managed to outperform humans on a majority of the ATARI games (Hessel et al., 2017; Mnih, Kavukcuoglu, et al., 2015).

Extrapolating this trend towards greater generality and intelligence, we may expect AI systems to eventually surpass humans on both these metrics. Such systems are sometimes called artificial general intelligences (AGIs). Indeed, many expect the creation of the first human-level AGI to occur within the next few decades, and greatly superhuman AGI soon after (Bostrom, 2014; and Section 2.2 below).

[1] This introduction shares some material with Everitt and Hutter (submitted 2018) and Everitt, Lea, et al. (2018).

Controlling AGI. A superhuman AGI is a system that outperforms humans on most cognitive tasks. In order to control it, humans would need to control a system more intelligent than themselves. This may be nearly impossible if the difference in intelligence is large, and the AGI is trying to escape control.

Humans have one key advantage: As the designers of the system, we get to decide the AGI's goals, and the way the AGI strives to achieve its goals.[2] This may allow us design AGIs whose goals are *aligned* with ours, and that pursue them in a responsible way. Increased intelligence in an AGI is not a threat as long as the AGI only strives to help us achieve our own goals.

On a high level, this *goal alignment problem* is challenging for at least two reasons: Specificity of human values, and Goodhart's law. First, human values are rather specific. For example, most humans prefer futures with happy sentient beings over futures with much suffering or no sentient beings at all. But even futures where happiness greatly outweighs suffering can still be undesirable. Yudkowsky (2009) gives an example where a happy, sentient experience is set to repeat everywhere until the heat-death of the universe. Most people find it repulsive, because it lacks important values such as variation and growth. However, listing *all* important values is not an easy task, which makes it hard to define a goal for an AGI that matches our human values even if the goal is optimized by a highly intelligent system. In this way, an AGI resembles a fairy-tale genie who grants you a wish, but interprets it over-literally (Wiener, 1960). In most fairy tales, the hero eventually wants his wish undone.

A second reason why alignment is hard is Goodhart's law (Goodhart, 1975, 1984), which roughly states that:

> "When a measure becomes a target, it ceases to be a good measure." (Strathern, 1997).

In the context of AGI, it means that if we give our system a proxy-measure for how well it is satisfying a goal, then the proxy is likely to cease being a good measure once the AGI starts optimizing for it. An oft-mentioned example is that of a reinforcement learning (RL) agent that uses a reward signal as goal proxy. In most circumstances, the reward may correspond well to the agent's performance. But an AGI optimizing the reward may find a way to short circuit the reward signal, obtaining maximum reward for behaviors considered undesirable by its human designers (Ring and Orseau, 2011). This is a rather extreme instance of Goodhart's law. Section 4.5 of this thesis is devoted to various aspects of the goal alignment problem.

[2]The goal does not need to be a long-term goal (Armstrong, Sandberg, et al., 2012; Christiano, 2015).

Other aspects. In addition to goal alignment, some further properties may also contribute to making an AGI safe. We may for example ask that the AI system is *corrigible* (Soares, Fallenstein, et al., 2015) in the sense that it lets us correct mistakes in its source that we have made while designing it, and that it lets us shut it down if we judge that to be the best option. Unfortunately, many AI designs give the agent an incentive to prevent corrections and shut down. This, of course, is a type of misalignment. There has been some debate about whether corrigibility should be treated as a property separate from goal alignment (Carey, 2018), or be made to follow from a more general approach to goal alignment (Hadfield-Menell, Dragan, et al., 2017; Milli et al., 2017). We will study some aspects of corrigibility in Chapter 11.

The way an AGI makes decisions is also important. Even if its goals are aligned, it may still do the wrong thing if it follows the wrong decision principle. While expected utility maximization is a principle that many agree on, there are subtleties in how the expectation should be computed when the agent is part of the environment that it is interacting with. Some aspects of decision theory will be considered in Section 6.4 and Chapter 10.

Why study AGI before it exists? Why study the safety of AGI before it exists, and before we even know whether it will ever exist? There are at least two types of reasons for this. The first is pragmatic. If AGI is created, and we do not know how to control it, then the outcome could be catastrophic (Bostrom, 2014). It is customary to take precautions not only against catastrophes we know will happen, but also against catastrophes that have only a slight chance of occurring (for example, a city may decide to build earthquake safe buildings, even if the probability of an earth quake occurring is fairly low). As discussed in Section 2.2 below, many believe that AGI has more than a small probability of occurring reasonably soon, and it can potentially cause very significant catastrophes.

The second reason is scientific. Potential future AGIs are theoretically interesting objects, and the question of how a human can control machines more intelligent than him or herself is philosophically stimulating.

Main contributions. The main contributions of this thesis are:

- A formal definition of alignment (Section 5.4).
- A method for detecting misalignment problems in RL setups (Section 5.5).
- Formalization and categorization of misalignment problems in three different RL setups (Chapters 6 to 8).

- Descriptions of a number of high-level *tools* for managing misalignment, as well as identification of which tools solve which type of misalignment problem (Chapters 6 to 9).
- Extensions of the two standard physicalistic decision theories (causal and evidential decision theory) to the sequential case (Section 10.2).
- A characterization of the robot's best action in the off-switch game for arbitrary belief distribution and irrationality assumptions (Chapter 11).

Outline. The thesis is divided into three parts. The first part describes the relevant background:

Chapter 2: A growing literature is slowly starting to make progress on how to think about future AGIs and how to make them safe. This chapter provides a literature review for this emerging field.

Chapter 3: In the early days of AI research, a wide range of topics were considered. The last two decades have seen AI research converge on *rational agents* that act to achieve some goal (Russell and Norvig, 2010). Chapter 3 gives a gentle introduction to universal artificial intelligence (UAI), which is a formal theory for rational agents making only very weak assumptions about the environment they are in.

Chapter 4: Causal graphs elegantly combine the power of formal mathematics and probability theory with intuitive visualizations. They will be used extensively in almost every subsequent chapter. This chapter describes the basics of causal graphs, as well as some of our own notation.

The second part contains my work related to the alignment problem of rational agents:

Chapter 5: As mentioned, goal alignment is a central concern for AGI safety. In order to study alignment formally, Chapter 5 extends the UAI framework from Chapter 3 in several ways, formally defines alignment, and proposes a method for detecting misalignment in RL setups.

Chapters 6 to 8: Different real-world applications of RL differ in whether the reward is provided by a preprogrammed reward function, a human, or by an interactively learned reward function. The distinction gives rise to important differences in alignment problems. Chapters 6 to 8 study misalignment

problems and ways they can be avoided or reduced in the three different RL setups.

Chapter 9: One source of misalignment is the AGI corrupting its reward signal. Chapter 9 formalizes reward corruption in a Markov decision process, and derives several positive and negative results for how and when the problem can be mitigated.

The third part contains chapters on decision theory and corrigibility:

Chapter 10: The questions of how to calculate expected utility and make decisions become surprisingly complex when the agent is part of the environment that it interacts with. Causal and evidential decision theory are the two most popular decision theories among modern philosophers. This chapter extends them to the sequential case, where multiple actions and observations are interleaved.

Chapter 11: Being able to shut down an AGI is an important safety feature. However, the agent shutting down after receiving a request to do so may not always represent the human's best interests. This chapter deepens the analysis of a previously proposed model of this problem.

Finally, Chapter 12 concludes with summary and discussion. The reader may also want to occasionally consult the list of notation in Appendix A at the end of this thesis.

In some chapters, semi-formal *statements* are made in place of fully formal theorems. This is to improve the flow of the text, and to avoid readers getting stuck on inessential details. These statements are always backed by fully formal theorems that are either referenced or found in appendices at the end of the chapter.

"Within thirty years, we will have the technological means to create superhuman intelligence. Shortly after, the human era will be ended."

Vernor Vinge (1993)

2. AGI Safety Literature Review[1]

This chapter reviews the literature relevant to AGI safety. An extensive survey of the AGI safety literature was previously made by Sotala and Yampolskiy (2014). Since then, the field has grown significantly. More up-to-date references are provided by this chapter, and by a number of recent research agendas and problem collections (Amodei, Olah, et al., 2016; Leike, Martic, et al., 2017; Russell, Dewey, et al., 2016; Soares and Fallenstein, 2017; Stoica et al., 2017; Taylor et al., 2016). A recent inventory of AGI projects and their attitudes towards ethics and safety also contributes to an overview of AGI safety research and attitudes (Baum, 2017a).

This thesis is structured as follows. Progress on how to think about yet-to-be-designed future AGI's is described in the first section (Section 2.1). Based partly on this understanding, we next survey predictions for when AGI will be created and what will happen after its creation (Section 2.2). We list and discuss identified AGI safety problems (Section 2.3), as well as proposals for solving or mitigating them (Section 2.4). Finally, we review the current public policy on AGI safety issues (Section 2.5), before making some concluding remarks (Section 2.6).

2.1. Understanding AGI

A major challenge for AGI safety research is to find the right conceptual models for plausible AGIs. This is especially challenging since we can only guess at the technology, algorithms, and structure that will be used. Indeed, even if we had the blueprint of an AGI system, understanding and predicting its behavior might still be hard: Both its design and its behavior could be highly complex. Nonetheless, several abstract observations and predictions are possible to make already at this stage.

2.1.1. Defining Intelligence

Legg and Hutter (2007c) propose a formal definition of intelligence based on algorithmic information theory and the UAI framework (Chapter 3; Hutter, 2005). They compare

[1] This chapter is based on Tom Everitt, Gary Lea, and Marcus Hutter (2018). "AGI Safety Literature Review". In: *International Joint Conference on Artificial Intelligence (IJCAI)*.

it to a large number of previously suggested definitions (Legg and Hutter, 2007a). Informally, their definition states that:

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

The definition is non-anthropomorphic, meaning that it can be applied equally to humans and artificial agents. All present-day AIs are less intelligent than humans according to this definition, as each AI is unable to achieve goals beyond a rather narrow domain. These domains can be for example ATARI environments (Hessel et al., 2017; Mnih, Badia, et al., 2016; Mnih, Kavukcuoglu, et al., 2015), board-games (Silver, Huang, et al., 2016; Silver, Hubert, et al., 2017; Silver, Schrittwieser, et al., 2017), car-driving (Bojarski et al., 2016; Huval et al., 2015). However, as discussed in Chapter 1, a trend towards greater generality can be observed.

Following the Legg-Hutter definition, we may expect that a future, super-human AGI will be able to achieve more goals in a wider range of environments than humans. The most intelligent agent according to this definition is AIXI, which has been studied both mathematically and empirically; see Chapter 3, Everitt and Hutter (2018), Hutter (2005, 2012b), and Leike (2016) for surveys. Most of this thesis is based around the UAI framework. Other safety work in the UAI framework is reviewed mostly in Section 2.4.

The Legg-Hutter intelligence definition measures what matters for control. The more intelligent an agent is, the more control it will have over aspects of the environment relating to its goals. If two agents with significantly different Legg-Hutter intelligence have conflicting goals in a shared environment, then in many environments we may expect the more intelligent of the two to succeed and the less intelligent fail. This points to the risks with increasingly intelligent AGIs: If their goals are not aligned with ours, then there will likely be a point where their goals will be achieved to the loss of ours (Russell, 2016).

2.1.2. Orthogonality

Bostrom's (2012, 2014) *orthogonality thesis* states that essentially any level of intelligence is compatible with any type of goal. Thus it does not follow, as is sometimes believed, that a highly intelligent AGI will realize that a simplistic goal such as creating paperclips or computing decimals of π is dumb, and that it should pursue something more worthwhile such as art or human happiness. Relatedly, Hume (1738) argued that *reason* is the slave of *passion*, and that a passion can never rationally be derived. In other words, an AGI will employ its intelligence to achieve its goals, rather than conclude

that its goals are pointless. Further, if we want an AGI to pursue goals that we approve of, we better make sure that we design the AGI to pursue such goals: Beneficial goals will not emerge automatically as the system gets smarter.

2.1.3. Convergent Instrumental Goals

The orthogonality thesis holds for the *end goals* of the system. In stark contrast, the *instrumental goals* will often coincide for many agents and end goals (Bostrom, 2012; Omohundro, 2007, 2008). Common instrumental goals include:

- Self-improvement: By improving itself, the agent becomes better able at achieving its end goal.
- Goal-preservation and self-preservation: By ensuring that future versions of itself pursues the same goals, the end goal is more likely to be achieved.
- Resource acquisition: With more resources, the agent will be better able at achieve the end goals.

Exceptions exists, especially in game-theoretic situations where the actions of other agents may depend on the agent's goals or other properties (Lavictoire et al., 2014). For example, an agent may want to change its goals so that it always chooses to honor contracts. This may make it easier for the agent to make deals with other agents. The sequential toxoplasmosis problem can also be seen as a case of this (Example 10.5 in Chapter 10).

2.1.4. Formalizing AGI

Bayesian, history-based agents have been used to formalize AGI in the UAI framework (Chapter 3; Hutter, 2005). Extensions of this framework have been developed for studying multi-agent interaction (Leike, Taylor, et al., 2016), space-time embeddedness (Orseau and Ring, 2012), self-modification (Everitt, Filan, et al., 2016; Orseau and Ring, 2011), observation modification (Ring and Orseau, 2011), self-duplication (Orseau, 2014a,b), knowledge seeking (Orseau, 2014c), decision theory (Everitt, Leike, et al., 2015), and others (Everitt and Hutter, 2018).

Some aspects of reasoning are swept under the rug by AIXI and Bayesian optimality. Importantly, probability theory assumes that agents know all the logical consequences of their beliefs (Gaifman, 2004). An impressive model of *logical non-omniscience* has recently been developed by Garrabrant et al. (2016, 2017). Notably, Garrabrant's theory avoids Gödelian obstacles for agents reasoning about improved versions of themselves

(Fallenstein and Soares, 2014). There is also hope that it can provide the foundation for a decision theory for logically uncertain events, such as how to bet on the 50th digit of π before calculating it.

2.1.5. Alternate Views

Eric Drexler (private communication, 2017) argues that an AGI does not need to be an *agent* that plans to achieve a goal. An increasingly automatized AI research and development process where more and more of AI development is being performed by AI tools can become super-humanly intelligent without having any agent subcomponent. Avoiding to implement goal-driven agents that make long-term plans may avoid some safety concerns. However, Bostrom (2014, Ch. 10) worries that even *tool AIs* that are not explicitly agents may still obtain some agent-like properties as their intelligence increases. Constructing an agent where the goals can be set explicitly may therefore be a safer option, though Drexler (2015) has an idea for how to keep AIs specialized. A drive for creating agent AIs rather than tool AIs can also arise from economic incentives (Gwern, 2016).

Relatedly, Weinbaum and Veitas (2016) criticize the (rational) agent assumption underpinning most AGI theory.

2.2. Predicting AGI Development

Based on historical observations of economical and technological progress, and on the growing understanding of potential future AGIs described in Section 2.1, predictions have been made both for when the first AGI will be created, and what will happen once it has been created.

2.2.1. When Will AGI Arrive?

There is an ongoing and somewhat heated debate about when we can expect AGI to be created, and whether AGI is possible at all or will ever be created. For example, by extrapolating various technology trends until we can emulate a human brain, Kurzweil (2005) argues that AGI will be created around 2029. Chalmers (2010) makes a more careful philosophical analysis of the brain-emulation argument for AI, and shows that it defeats and/or avoids counter arguments made by Dreyfus (1972), Lucas (1961), Penrose (1994), and Searle (1980). Chalmers is less optimistic about the timing of AGI, and only predicts that it will happen within this century.

Surveys of when AI researchers estimate that human-level AGI will be created have been made by Baum et al. (2011), V. C. Müller and Bostrom (2016), and Grace et al. (2017). Baum et al. (2011) asked 21 attendees at the 2009 Artificial General Intelligence conference, and found a median of 2045 for superhuman AGI. V. C. Müller and Bostrom (2016) made a bigger survey of 550 people from the 2012 Philosophy and Theory of AI (PT-AI) and AGI conferences, the Greek Society for Artificial Intelligence (EETN), as well as the top 100 most cited authors in artificial intelligence. The medians for the various groups all fell between 2040 and 2050. Grace et al. (2017) got 352 responses from NIPS and ICML 2015 presenters on the slightly different question of when AGI will accomplish all tasks better than human workers, and got a median of 2061. Interestingly, Asian respondents predicted AGI more than 30 years sooner than North American respondents, with Europeans in the middle slightly closer to the Asians than to the North Americans. It is also worth noting that estimates vary widely, from never to just a few years into the future.

There are also other indicators of when AGI might arrive. Algorithmic progress have been tracked by Grace (2013), Eckersley and Nasser (2018), and AI Impacts (2018b), and the costs of computing have been tracked by AI Impacts (2018a). A new MIT course on AGI shows that the idea of AGI approaching is becoming more mainstream (Fridman, 2018). Stanford has a course on AI safety (Sadigh, 2017). Jilk (2017) argues that an AGI must have a conceptual-linguistic faculty in order to be able to access human knowledge or interact effectively with the world, making a conceptual-linguistic faculty a necessary requirement for AGI. Jilk further argues that a conceptual-linguistic faculty is likely to be a strong indicator of AGI being near, and suggests ways in which we may test whether a system has a conceptual-linguistic ability.

2.2.2. Will AGI Lead to a Technological Singularity?

As explained in Section 2.1.3, one of the instrumental goals of almost any AGI will be self-improvement. The greater the improvement, the likelier the end goals will be achieved. This can lead to *recursive* self-improvement, where a self-upgraded AGI is better able to find yet additional upgrades, and so on. If the pace of this process increases, we may see an *intelligence explosion* once a critical level of self-improvement capability has been reached (Bostrom, 2014; Good, 1966; Hutter, 2012a; Kurzweil, 2005; Vinge, 1993; Yudkowsky, 2008b). Already John von Neumann has been quoted calling this intelligence explosion a *singularity* (Ulam, 1958). Singularity should here not be understood in its strict mathematical sense, but more loosely as a point where our models break.

Some counter arguments to the singularity have been structured by Walsh (2016),

who argues that an intelligence explosion is far from inevitable:

- Intelligence measurement: The singularity predicts an increasingly rapid development of intelligence. However, it is not quite clear how we should measure intelligence (Hutter, 2012a). A rate of growth that looks fast or exponential according to one type of measurement, may look ordinary or linear according to another measurement (say, the log-scale).

- Fast thinking dog: No matter how much we increase the speed at which a dog thinks, the dog will never beat a decent human at chess. Thus, even if computers keep getting faster, this alone does not entail their ever becoming smarter than humans.

- Anthropocentric: Proponents of the singularity often believe that somewhere around the human level of intelligence is a critical threshold, after which we may see quick recursive self-improvement. Why should the human level be special?

- Meta-intelligence, diminishing returns, limits of intelligence, computational complexity: It may be hard to do self-improvement or be much smarter than humans due to a variety of reasons, such as a fundamental (physical) upper bound on intelligence or difficulty of developing machine learning algorithms.

These arguments are far from conclusive, however. In *Life 3.0*, Tegmark (2017) argues that AGI constitutes a third stage of life. In the first stage, both hardware and software is evolved (e.g. in Bacteria). In the second stage, the hardware is evolved but the software is designed. The prime example is a human child who goes to school and improves her knowledge and mental algorithms (i.e. her software). In the third stage of life, both the software and hardware is designed, as in an AGI. This may give unprecedented opportunities for quick development, countering the anthropocentric argument by Walsh. In relation to the fast thinking dog and the limits of intelligence arguments, Bostrom (2014) argues that an AGI may think up to a million times faster than a human. This would allow it to do more than a millennium of mental work in a day. Such a speed difference would make it very hard for humans to control the AGI. Powerful mental representations may also allow an AGI to quickly supersede human intelligence in quality Sotala (2017). Yampolskiy (2017) also replies to Walsh's arguments.

Kurzweil's (2005) empirical case for the singularity has been criticized for lack of scientific rigor (Modis, 2006). Modis (2002) argues that a logistic function fits the data better than an exponential function, and that logistic extrapolation yields that the rate of complexity growth in the universe should have peaked around 1990.

In conclusion, there is little consensus on whether and when AGI will be created, and what will happen after its creation. Anything else would be highly surprising, given that no similar event have previously occurred. Nonetheless, AGI being created within the next few decades and quickly superseding human intelligence seems like a distinct possibility.

2.2.3. Risks Caused by AGI

A technological singularity induced by an AGI may lead to existential risks such as a paperclip maximizer annihilating life in pursuit of resources (Bostrom, 2013, 2014), as well as risks of substantial suffering without extinction (Sotala and Gloor, 2017). However, even if AGI does not lead to a technological singularity, it may still cause substantial problems, for example through (enabling greater degrees of) social manipulation, new types of warfare, or shifts in power dynamics (Sotala, 2018). Categorizations of possible scenarios have been proposed by Turchin (2018) and Yampolskiy (2016).

2.3. Problems with AGI

Several authors and organizations have published research agendas that identify potential problems with AGI. Russell, Dewey, et al. (2016) and the Future of Life Institute (FLI) take the broadest view, covering societal and technical challenges in both the near and the long term future. Soares and Fallenstein (2014, 2017) at the Machine Intelligence Research Institute (MIRI) focus on the mathematical foundations for AGI, including decision theory and logical non-omniscience. Several subsequent agendas and problem collections try to bring the sometimes "lofty" AGI problems down to concrete machine learning problems: Amodei, Olah, et al. (2016) at OpenAI et al., Leike, Martic, et al. (2017) at DeepMind, and Taylor et al. (2016) also at MIRI. In the agenda by Stoica et al. (2017) at UC Berkeley, the connection to AGI has all but vanished. For brevity, we will refer to the agendas by the organization of the first author, with MIRI-AF the agent foundations agenda by Soares and Fallenstein (2014, 2017) and MIRI-ML the machine learning agenda by (Taylor et al., 2016). Figure 2.1 shows some connections between the agendas. Figure 2.1 also makes connections to research done by other prominent AGI safety institutions Oxford Future of Humanity Institute (FHI), Australian National University (ANU), and Center for Human-Compatible AI (CHAI).

Some clusters of problems appear in multiple research agendas:

- Value specification: How do we get an AGI to work towards the right goals? MIRI calls this value specification. Bostrom (2014) discusses this problem at length,

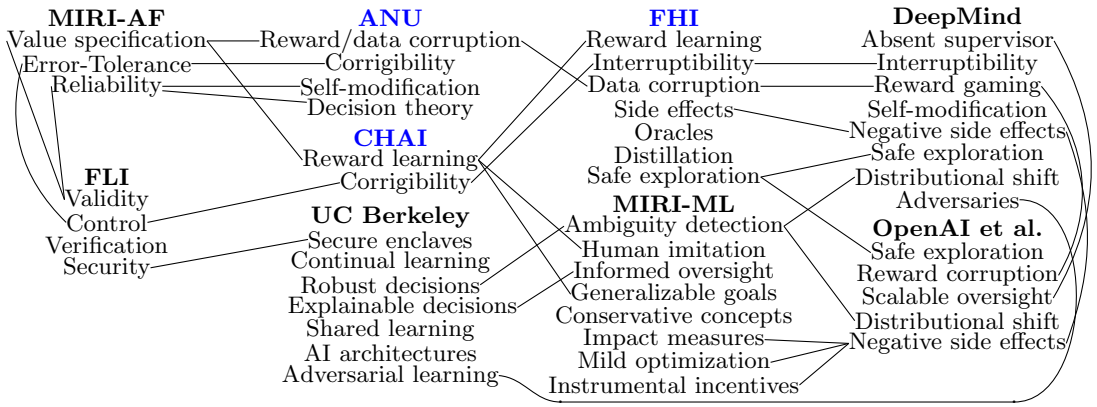


Figure 2.1.: Connections between problems stated in different AGI safety research agendas (for ANU, CHAI, and FHI, the agendas are inferred from their recent publications).

arguing that it is much harder than one might naively think. Davis (2015) criticizes Bostrom's argument, and Bensinger (2015) defends Bostrom against Davis' criticism. Reward corruption, reward gaming, and negative side effects are subproblems of value specification highlighted in the DeepMind and OpenAI agendas.

- Reliability: How can we make an agent that keeps pursuing the goals we have designed it with? This is called *highly reliable agent design* by MIRI, involving decision theory and logical omniscience. DeepMind considers the self-modification subproblem.

- Corrigibility: If we get something wrong in the design or construction of an agent, will the agent cooperate in us trying to fix it? This is called error-tolerant design by MIRI-AF and *corrigibility* by Soares, Fallenstein, et al. (2015). The problem is connected to safe interruptibility as considered by DeepMind.

- Security: How to design AGIs that are robust to adversaries and adversarial environments? This involves building sandboxed AGI protected from adversaries (Berkeley), and agents that are robust to adversarial inputs (Berkeley, DeepMind).

- Safe learning: AGIs should avoid making fatal mistakes during the learning phase. Subproblems include safe exploration and distributional shift (DeepMind, OpenAI), and continual learning (Berkeley).

- Intelligibility: How can we build agent's whose decisions we can understand? Explainable decisions (Berkeley); informed oversight (MIRI). DeepMind is also working on these issues, see Section 2.4.5 below.

- Societal consequences: AGI will have substantial legal, economic, political, and military consequences. Only the FLI agenda is broad enough to cover these issues, though many of the mentioned organizations evidently care about the issue (Brundage et al., 2018; DeepMind, 2017).

There are also a range of less obvious problems, which have received comparatively less attention:

- Subagents: An AGI may decide to create *subagents* to help it with its task Orseau (2014a,b) and Soares, Fallenstein, et al. (2015). These agent's may for example be copies of the original agent's source code running on additional machines. Subagents constitute a safety concern, because even if the original agent is successfully shut down, these subagents may not get the message. If the subagents in turn create subsubagents, they may spread like a viral disease.

- Malign priors: Christiano (2016) argues that the *universal prior* M (see Section 3.4) is malign, as it is likely to be dominated by hypotheses that contain agents with agendas to influence the user of the prior. While it is unclear to what extent this type of problem would affect any practical agent, it bears some semblance to aggressive *memes*, which do cause problems for human reasoning (Dennett, 1990).

- Physicalistic decision making: The *rational agent* framework is pervasive in the study of artificial intelligence. It typically assumes that a well-delineated entity interacts with an environment through action and observation channels. This is not a realistic assumption for *physicalistic* agents such as robots that are part of the world they interact with (Soares and Fallenstein, 2014, 2017). Chapter 10 considers this question.

- Indeterminate agency: An artificial intelligence may be copied and distributed, allowing instances of it to interact with the world in parallel. This can significantly boost learning, but undermines the concept of a single agent interacting with the world.

- Meta-cognition: Agents reasoning about their own computational resources and *logically uncertain* events can encounter strange paradoxes due to Gödelian limitations (Fallenstein and Soares, 2015; Soares and Fallenstein, 2014, 2017) and shortcomings of probability theory (Soares and Fallenstein, 2014, 2015a, 2017).

While these problems may seem esoteric, a security mindset (Yudkowsky, 2017) dictates that we not only protect ourselves from things that can clearly go wrong, but also against

anything that is not guaranteed to go right. Indeed, unforeseen errors often cause the biggest risks. For this reason, the biggest safety problem may be one that we have not thought of yet – not because it would necessarily be hard to solve, but because in our ignorance we will fail to adopt measures to mitigate the problem.

2.4. Design Ideas for Safe AGI

We next look at some ideas for creating safe AGI. There is not always a clear line distinguishing ideas for safe AGI from other AI developments. Many works contribute to both simultaneously.

2.4.1. Value Specification

RL and misalignment. Reinforcement learning (RL) (Sutton and Barto, 1998) is currently the most promising framework for developing intelligent agents and AGI. Combined with Deep Learning, it has seen some remarkable recent successes, especially in playing board games (Silver, Huang, et al., 2016; Silver, Hubert, et al., 2017; Silver, Schrittwieser, et al., 2017) and computer games (Hessel et al., 2017; Mnih, Badia, et al., 2016; Mnih, Kavukcuoglu, et al., 2015).

Aligning the goals of an RL agent with the goals of its human supervisor comprise significant challenges, however. These challenges include correct specification of the reward function, and avoiding that the agent takes shortcuts in optimizing it. Such shortcuts include the agent corrupting the observations on which the reward function evaluates performance, modifying the reward function to give more reward, hijacking the reward signal or the memory location of the reward, and, in the case of an interactively learned reward function, corrupting the data training the reward function. Chapters 5 to 8 categorize misalignment problems in RL, and suggest a number of techniques for managing the various sources of misalignment. The rest of this subsection reviews other work that has been done on designing agents with correctly specified values.

Learning a reward function from actions and preferences. One of the main challenges in scaling RL to the real world includes designing the reward function. This is particularly critical for AGI, as a poorly designed reward function would likely lead to a misaligned agent. As an example of misalignment, Clark and Amodei (2016) found that their boat racing agent preferred going in circles and crashing into obstacles instead of winning the race, due to a subtly misspecified reward function. Gwern (2011), Irpan (2018), and Lehman et al. (2018) have many more examples. Analogous failures in AGIs

could cause severe catastrophes. The DeepMind problem collection calls this a *reward gaming* problem. One potential way around the problem of gameable reward functions is to let the agent learn the reward function. This lets designers offload some of the design work to powerful machine learning techniques.

Inverse reinforcement learning (IRL) (Choi and Kim, 2011; Ng and Russell, 2000; Ziebart et al., 2008) is a framework for learning a reward function from the actions of an expert, often a human demonstrator. In one famous example, Abbeel et al. (2007) taught an agent acrobatic helicopter flight by observing the actions of a human pilot. Impressively, the agent ultimately became better at flying than the pilot it observed. However, a learned reward function cannot be better than the data that trained it. If all training happens before the agent is launched into the environment, then the data may not properly describe situations that the agent reaches far into its lifetime (a so-called *distributional shift* problem; Amodei, Olah, et al., 2016). For this reason, interactive training of the reward function may be preferable, as it allows the training data to adapt to any new situation the agent may encounter.

Cooperative inverse reinforcement learning (CIRL) is a generalization of IRL that lets the expert and the agent act simultaneously in the same environment, with the agent interactively learning the expert's preferences (Hadfield-Menell, Dragan, et al., 2016). Among other things, this allows the expert to take demonstrative actions that are suboptimal according to his or her reward function but more informative to the agent, without the agent being led to infer an incorrect reward function. The CIRL framework can be used to build agents that avoid interpreting reward functions overly literally, thus avoiding some misalignment problems with RL (Hadfield-Menell, Milli, et al., 2017).

A reward functions can also be learned from a human rating short video clips of (partial) agent trajectories against each other (Christiano et al., 2017). For example, if the human consistently rates scenarios where the agent falls off a cliff lower than other scenarios, then the learned reward function will assign a low reward to falling off a cliff. Using this technique, a non-expert human can teach an agent complex behaviors that would have been difficult to directly program a reward function for. Warnell et al. (2017) use a related approach, needing only 15 minutes of human feedback to teach the agent the ATARI game Bowling.

On a fundamental level, learning from actions and learning from preferences is not widely different. Roughly, a choice of action a over action b can be interpreted as a preference for the future trajectories resulting from action a over the trajectories resulting from action b. However, a few notable differences can still be observed. First, at least in Christiano et al.'s (2017) framework, preferences always apply to *past* events. In contrast,

an action in the CIRL framework typically gives information about which *future* events the human prefers. A drawback is that in order for the action to carry information about future events, the action must be chosen (somewhat) rationally. Humans do not always act rationally; indeed, we exhibit several systematic biases (Kahneman, 2011). A naive application of (C)IRL therefore runs the risk of inferring an incorrect reward function. To address this, Evans et al. (2016) develop a method for learning the reward function of agents exhibiting some human-like irrationalities. Without assumptions on the type of irrationality the expert exhibits, nothing can be learned about the reward function (Armstrong and Mindermann, 2017). In comparison, learning from preferences seems to require weaker rationality assumptions on the human's part, as correctly stating one's preferences may be easier than acting rationally.

Yet another approach to learning a reward function is to learn it from stories (Riedl and Harrison, 2016).

Approval-directed agents. In a series of blog posts, Christiano (2014) suggests that AGIs should be designed to maximize approval for their actions rather than trying to reach some goal. He argues that approval-directed systems have many of the same benefits of goal-directed systems while avoiding some of their worst pitfalls. Christiano (2015) and Cotra (2018) outline a method for how approval-directed agents can be chained together in a hierarchy, boosting the accuracy of the approvals of the human at the top of the chain.

Reward corruption. Reinforcement learning AGIs may hijack their reward signal and feed themselves maximal reward (Ring and Orseau, 2011). Interestingly, model-based agents with preprogrammed reward functions are much less prone to this behavior (Everitt, Filan, et al., 2016; Hibbard, 2012). However, if the reward function is learned online as discussed above, it opens up the possibility of *reward learning corruption*. An AGI may be tempted to influence the data training its reward function so it points towards simple-to-optimize reward functions rather than harder ones (Armstrong, 2015). Everitt, Krakovna, et al. (2017) (Chapter 9 in this thesis) show that the type of data the agent receives matter for reward learning corruption. In particular, if the reward data can be cross-checked between multiple sources, then the reward corruption incentive diminishes drastically. Everitt, Krakovna, et al. also evaluate a few different approaches to reward learning, finding that the *human action*-data provided in CIRL is much safer than the *reward*-data provided in standard RL, but that CIRL is not without worrying failure modes.

Side effects. An AGI that becomes overly good at optimizing a goal or reward function that does not fully capture all human values, may cause significant negative side effects (Yudkowsky, 2009). The *paperclip maximizer* that turns the earth and all humans into paperclips is an often used example (Bostrom, 2014), now available as a highly addictive computer game (Lantz, 2017). Less extreme examples include companies that optimize profits and cause pollution and other externalities as negative side effects.

The most serious side effects seem to occur when a target function is optimized in the extreme (such as turning the earth into paperclips). Quantilization can avoid over-optimization under some assumptions (Everitt, Krakovna, et al., 2017; Taylor, 2016). Another more specific method is to "regularize" reward by the impact the policy is causing (Armstrong and Levinstein, 2017). How to measure impact remains a major open question, however.

Morality. Both game-theory (Letchford et al., 2008) and machine learning (Conitzer et al., 2017; Shaw et al., 2018) have been suggested as ways to endow AI systems with a sense of morality. For example, Kleiman-Weiner et al. (2017) use hierarchical Bayesian inference to infer a moral theory from actions. However, Bogosian (2017) argues that a potential problem with these approaches is that the agent becomes over-confident in a hypothesis – for example due to a model-class that only represents a subset of possible moral theories. Bogosian therefore argues that it is better to build an agent that remains uncertain about which moral theory is correct, instead of building an agent with a single moral theory, be it learned or not. Agents that do not remain uncertain may commit big atrocities as judged by theories they have discarded, which is undesirable for believers in the discarded moral theory. In contrast, morally uncertain agents avoid events that are extremely bad according to any (possible) moral theory.

Virtue ethics has recently re-emerged as a third contender to consequentialist (utilitarian) and deontological (rule-based) ethics. Murray (2017) makes a case for building moral AI systems based on (Stoic) virtues rather then consequentialist reward-maximization.

Connections to economics. The goal alignment problem has several connections to the economics literature. It may be seen as an instance of Goodhart's law (Goodhart, 1975), which roughly states that any measure of performance ceases to be a good measure once it is optimized for. Manheim and Garrabrant (2018) categorize instances of Goodhart's law. It may also be seen as a principal-agent problem: The connections have been fleshed out by Hadfield-Menell and Hadfield (2018).

2.4.2. Reliability

Self-modification. Even if the reward function is correctly specified, an AGI may still be able to corrupt either the reward function itself or the data feeding it. This can happen either intentionally if such changes can give the agent more reward, or accidentally as a side effect of the agent trying to improve itself (Section 2.1.3).

A utility self-preservation argument going back to at least Schmidhuber (2007) and Omohundro (2008) says that agents should not want to change their utility functions, as that will reduce the utility generated by their future selves, as measured by the current utility function. Everitt, Filan, et al. (2016) formalize this argument, showing that it holds under three non-trivial assumptions: (1) The agent needs to be model-based, and evaluate future scenarios according to its current utility function; (2) the agent needs to be able to predict how self-modifications affect its future policy; and (3) the reward function itself must not endorse self-modifications. In RL (Sutton and Barto, 1998), model-free agents violate the first assumption, off-policy agents such as Q-learning violate the second, and the third assumption may fail especially in learned reward/utility functions (Section 2.4.1). Hibbard (2012) and Orseau and Ring (2011) also study the utility self-preservation argument.

Decision theory. Functional decision theory consolidates years of research about decision theory for embedded agents. Yudkowsky and Soares (2017) argue that functional decision theory overcomes weaknesses of both evidential and causal decision theory. Chapter 10 of this thesis study sequential extensions of causal and evidential decision theory.

2.4.3. Corrigibility

By default, agents may resist shutdown and modifications due to the self-preservation drives discussed in Section 2.1.3. Three rather different approaches have been developed to counter the self-preservation drives.

Indifference. By adding a term or otherwise modifying the reward function, the agent can be made indifferent between some choices of future events, for example shutdown or software corrections (Armstrong, 2017b). For example, this technique can be used to construct variants of popular RL algorithms that do not learn to prevent interruptions (Orseau and Armstrong, 2016).

Ignorance. Another option is to construct agents that behave as if a certain event (such as shutdown or software modification) was certain not to happen (Everitt, Filan, et al., 2016). For example, off-policy agents such as Q-learning behave as if they will always act optimally in the future, thereby effectively disregard the possibility that their software or policy be changed in the future. Armstrong (2017b) show that ignorance is equivalent to indifference in a certain sense.

Uncertainty. In the CIRL framework (Hadfield-Menell, Dragan, et al., 2016), agents are uncertain about their reward function, and learn about the reward function through interaction with a human expert. Under some assumptions on the human's rationality and the agent's level of uncertainty, this leads to naturally corrigible agents (Hadfield-Menell, Dragan, et al., 2017; Wängberg et al., 2017). Essentially, the agent will interpret the human's act of shutting them down as evidence that being turned off has higher reward than remaining turned on. In some cases where the human is likely to make suboptimal choices, the agent may decide to ignore a shut down command. There has been some debate about whether this is a feature (Milli et al., 2017) or a bug (Carey, 2018). These aspects will be discussed further in Chapter 11.

Continuous testing. Arnold and Scheutz (2018) argue that an essential component of corrigibility is to detect misbehavior as early as possible. Otherwise, significant harm may be caused without available corrigibility equipment having been put to use. They propose a ethical testing framework that continually monitors the agent's behavior on simulated ethics tests.

2.4.4. Security

Adversarial counterexamples. Deep Learning (e.g. Goodfellow, Bengio, et al., 2016) is a highly versatile tool for machine learning, and a likely building block for future AGIs. Unfortunately, it has been observed that small perturbations of inputs can cause severe misclassification errors (Athalye et al., 2017; Evtimov et al., 2017; Goodfellow, Shlens, et al., 2014; Szegedy et al., 2013).

In a recent breakthrough, Katz et al. (2017) extend the Simplex algorithm to neural networks with rectified linear units (Relus). Katz et al. call the extended algorithm ReluPlex, and use it to successfully verify the behavior of neural networks with 300 Relu nodes in 8 layers. They gain insight into the networks' behaviors in certain important regions, as well as the sensitivity to adversarial perturbations.

2.4.5. Intelligibility

While it is infamously hard to understand exactly what a deep neural network has learned, recently some progress has been made. DeepMind's *Psychlab* uses tests from psychology implemented in a 3D environment to understand deep RL agents. The tests led them to a simple improvement of the UNREAL agent (Leibo et al., 2018). Zahavy et al. (2016) instead use the dimensionality reduction technique t-SNE on the activations of the top neural network layer in DQN (Mnih, Kavukcuoglu, et al., 2015), revealing how DQN represents policies in ATARI games. Looking beyond RL, Olah et al. (2017) summarize work on visualization of features in image classification networks in a beautiful Distill post. Another line of work tries to explain what text and speech networks have learned (Alvarez-Melis and Jaakkola, 2017; Belinkov and Glass, 2017; Lei et al., 2016).

2.4.6. Safe learning

During training, a standard Deep RL agent such as DQN commits on the order of a million catastrophic mistakes such as jumping off a cliff and dying (Saunders et al., 2017). Such mistakes could be very expensive if they happened in the real world. Further, we do not want an AGI to accidentally set off all nuclear weapons in a burst of curiosity late in its training phase. Saunders et al. (2017) propose to fix this by training a neural network to detect potentially catastrophic actions from training examples provided by a human. The catastrophe detector can then override the agent's actions whenever it judges an action to be too dangerous. Using this technique, they manage to avoid all catastrophes in simple settings, and a significant fraction in more complex environments. A similar ideas was explored by Lipton et al. (2016). Instead of using human-generated labels, their catastrophe detector was trained automatically on the agent's catastrophes. Unsurprisingly, this reduces but does not eliminate catastrophic mistakes. A survey over previous work on safe exploration in RL is provided by (García and Fernández, 2015).

2.4.7. Other

Oracles. Armstrong (2017a) and Armstrong, Sandberg, et al. (2012) argue that oracles that only answer questions may be more safe than other types of AGI. Sarma and Hay (2017) suggest that computer algebra systems are concrete instances of oracles suitable for further study.

Tripwires. Martin, Everitt, et al. (2016) use the AIXI framework to show that AGIs with rewards bounded to a negative range (such as $[-1, 0]$) will prefer their subjective

environment to end. In the AIXI framework, "death" always has an implicit reward of 0. This may lead such systems to self-destruct once they are intelligent enough to figure out how to do so. Thus, a negative reward range may be used as a type of *tripwire* (Bostrom, 2014), preventing superintelligent AGI before we are ready for it.

Homomorphic encryption. Trask (2017) shows how to train homomorphically encrypted neural networks. A potential application to AGI safety is to have a homomorphically encrypted AGI whose predictions and actions also come out encrypted. A human operator with the secret key can choose to decrypt them only when he wants to. This may make the system unintelligible to itself, and thereby potentially prevent certain types of self-modifications.

Boxing. A natural idea for keeping an AGI safe is to constrain its interaction with the real world. While it may be infeasible to constrain a highly intelligent AGI from breaking out of its constrains (Alfonseca et al., 2016; Yudkowsky, 2002), it may still be a useful technique for constraining more limited AGIs (Babcock et al., 2017).

Meta-cognition. Garrabrant et al.'s (2016, 2017) theory of logical induction leads to several theorems about systems reasoning about their own computations in a consistent manner, avoiding Gödelian and Löbian obstacles. Fallenstein and Kumar (2015) shows that systems of higher-order logic can learn to trust their own theorems, in a certain sense.

Weakening the agent concept. Hedden (2015) makes some progress on defining rational agency in cases where there is no clear entity making the decisions, shedding some light on the connection between personal identity and rational agency.

Physicalistic decision making. Recently, functional decision theory has been proposed as a decision theory that improves upon both evidential and causal decision theory (Yudkowsky and Soares, 2017).

Neuromorphic AGI. Jilk et al. (2017) argue that our best chance for creating safe AGI is to use the human brain as a blueprint, as it allows us to use our extensive knowledge about human drives to anticipate dangers and to develop a good training program for the agent. A common objection to this idea is that is very hard to mathematically prove properties about the human brain (Bostrom, 2014; Yudkowsky, 2008a). Countering this objection, Jilk et al. (2017) argue that it will likely be hard to prove properties

about any type of AGI. It is an open question whether any of the promising works we have reviewed above and develop in this thesis will lead to mathematical guarantees surpassing our potential trust in a neuromorphic AGI.

2.5. Public Policy on AGI

Recommendations. In a collaboration spanning more than a handful AGI safety-oriented organizations, Brundage et al. (2018) consider scenarios for how AI and AGI may be misused and give advice for both policy makers and researchers. Regulation of AI remains a controversial topic, however. On the one hand, Erdelyi and Goldsmith (2018) call for global regulatory body. Others worry that regulations may limit the positive gains from AI, and recommend increased public funding for safety research (Nota, 2015). Baum (2017b) is also wary of regulation, but for slightly different reasons. He argues that *extrinsic* measures such as regulations run the risk of backfiring, making AI researchers look for ways around the regulations. He argues that *intrinsic* measures that make AI researchers want to build safe AI are more likely to be effective, and recommends either purely intrinsic methods or combinations of intrinsic and extrinsic approaches. He lists some ideas for intrinsic approaches:

- Setting social norms and contexts for building safe AI, by creating and expanding groups of AI researchers that openly promote safe AI, and by having conferences and meetings with this agenda.
- Using "allies" such as car manufacturers (and the military) that want safe AI. AI researchers want to satisfy these organizations in order to access their funding and research opportunities.
- Framing AGI less as winner-take-all race, as a race implies a winner. Instead, it is important to emphasize that the result of a speedy race where safety is deprioritized is likely to be a very unsafe AGI. Also, framing AI researchers as people that are good (but sometimes do bad things), and emphasize that their jobs are not threatened by a focus on safety.
- Making researchers publicly state that they care about safe AI and then reminding them when they don't follow it can lead to cognitive dissonance, which may cause researchers to believe they want safe AI.

Armstrong, Bostrom, et al. (2016) counterintuitively find that information sharing between teams developing AGI exacerbates the risk of an AGI race.

Policy makers. Although public policy making is often viewed as the domain of public bodies, it should be remembered that many organizations such as corporations, universities and NGOs frequently become involved through advocacy, consulting, and joint projects. Indeed, such involvement can often extend to de facto or "private" regulation via organizational guidelines, organizational policies, technical standards and similar instruments.

Professional organizations have already taken a leading role. The IEEE, for example, is developing guidelines on Ethically Aligned Design (IEEE, 2017a,b). Meanwhile, the ACM and the SIGAI group of AAAI have co-operated to establish a new joint conference on AI, ethics and society, AIES (AI Matters, 2017). Economic policy and technical standards organizations have also started to engage: for example, the OECD has established a conference on "smart policy making" around AI developments (OECD, 2017) and ISO/IEC has established a technical committee on AI standards (ISO/IEC, 2017). Corporations and corporate consortia are also involved, typically through the public-facing aspects of their own corporate policies (IBM, 2018; Intel, 2017) or through joint development of safety policies and recommendations which consortia members will adopt (Partnership on AI, 2016).

Finally, in addition to the traditional public roles of academia and academics, there are an increasing number of academically affiliated or staffed AI organizations. With varying degrees of specificity, these work on technical, economic, social and philosophical aspects of AI and AGI. Organizations include the Future of Humanity Institute (FHI), the Machine Intelligence Research Institute (MIRI), the Centre for the Study of Existential Risk (CSER) and the Future of Life Institute (FLI).

Current policy anatomy. It could be said that public policy on AGI does not exist. More specifically, although work such as Baum (2017a) highlights the extent to which AGI is a distinct endeavor with its own identifiable risk, safety, ethics (RISE) issues, public policy AGI is currently seldom separable from default public policy on AI taken as a whole (PPAI). Existing PPAI are typically structured around (a) significant financial incentives (e.g. grants, public-private co-funding initiatives, tax concessions) and (b) preliminary coverage of ethical, legal and social issues (ELSI) with a view to more detailed policy and legislative proposals later on (FTI Consulting, 2018; Miller et al., 2018).

In the case of the EU, for example, in addition to experimental regulation with its new algorithmic decision-making transparency requirements in (EUR-lex, 2016, Article 22, General Data Protection Regulation) , its various bodies and their industry partners have committed over 3 billion Euro to AI and robotics R&D and engaged in two rounds

of public consultation on the European Parliament's proposed civil law liability framework for AI and robotics (Ansip, 2018). However, the much demanded first draft of an overarching policy framework is still missing, being slated for delivery by the European Commission no earlier than April 2018.

Elsewhere, spurred into action by the implications of the AlphaGo victory and China's recent activities (outlined below), South Korea and Japan have already rapidly commenced significant public and public-private investment programs together with closer co-ordination of state bodies, industry and academia (J. E. Ha, 2016; Volodzsko, 2017). Japan is also additionally allowing experimental regulation in some economic sectors (Takenaka, 2017). The UK has started work on a preliminary national policy framework on robotics and AI (Hall and Pesenti, 2017; UK Parliament, 2017), and have established a national Centre for Data Ethics and Innovation (CSER, 2017).

Current policy dynamics. Although there is substantial positive co-operation between universities, corporations and other organizations, there is a negative dynamic operating the nation-state, regional and international context. Contrary to the expert recommendations above, there is increasing rhetoric around an AI "arms race" (Cave and ÓhÉigeartaigh, 2018), typified by President Vladimir Putin's September 2017 comment that "... whoever becomes the leader in [the AI] sphere will become the leader in the world" (Apps, 2017). Relatedly, China's 8 July 2017 AI policy announcement included being the global leader in AI technology by 2030 (Ding, 2018; Kania, 2018; PRC State Council, 2017). It also included aims of "creating a safer, more comfortable and convenient society". Alongside this policy shift has been increased Sino-American competition for AI talent (Cyranoski, 2018). In the US, the Obama Administration began consultation and other moves towards an Federal policy framework for AI technology investment, development and implementation (Agrawal et al., 2016; White House OSTP, 2016). However, the Trump Administration abandoned the effort to focus mainly on military spending on AI and cyber-security (Metz, 2018).

Policy outlook. Given the above, looking forward it would appear that the organizations noted above will have to work hard to moderate the negative dynamic currently operating at the nation-state, regional and international level. Useful guidance for researchers and others engaging with public policy and regulatory questions on AI is given by 80 000 Hours (2017). Further references on public policy on AGI can be found in (Dafoe, 2017; Sotala and Yampolskiy, 2014).

2.6. Conclusions

AGI promises to be a major event for human kind. Recent research have made important progress on how to think about potential future AGIs, which enables us to anticipate and (hopefully) mitigate problems before they occur. This may be crucial, especially if the creation of a first AGI leads to an "intelligence explosion". Solutions to safety issues often have more near-term benefits as well, which further adds to the value of AGI safety research.

It is our hope that this summary will help new researchers enter the field of AGI safety, and provide traditional AI researchers with an overview of challenges and design ideas considered by the AGI safety community.

"Approximate the solution, not the problem."

Richard S. Sutton

3. Universal Artificial Intelligence[1]

Since the inception of the AI research field in the mid-twentieth century, a range of practical and theoretical approaches have been investigated. This chapter will discuss *universal artificial intelligence* (UAI) as a unifying framework and foundational theory for many of these approaches. The generality and formal power of UAI makes it an excellent framework for studying safety aspects of AGI. Many of the subsequent chapters will extend the UAI framework in different ways.

The chapter begins with some general background on the scientific study of intelligence and AI (Section 3.1). We then give an overview of the components of the UAI theory (Section 3.2), before considering each component in more depth: The framework (Section 3.3), learning (Section 3.4), goal formulation (Section 3.5), planning (Section 3.6), and the AIXI agent (Section 3.7). Some conclusions are provided (Section 3.8).

3.1. Background and History of AI

Intelligence is a fascinating topic, and has been studied from many perspectives. Cognitive psychology and behaviorism are psychological theories about how humans think and act. Neuroscience, linguistics, and the philosophy of mind try to uncover how the human mind and brain works. Machine learning, logic, and computer science can be seen as attempts to make machines that think.

Scientific perspectives on intelligence can be categorized based on whether they concern themselves with thinking or acting (cognitive science vs. behaviorism), and whether they seek objective answers such as in logic or probability theory, or try to describe humans as in psychology, linguistics, and neuroscience. The distinction is illustrated in Table 3.1. The primary focus of AI is on *acting* rather than *thinking*, and on *doing the right thing* rather than *emulating humans*. Ultimately, we wish to build systems that solve problems and act appropriately; whether the systems are inspired by humans or follow philosophical principles is only a secondary concern.

[1] This chapter is based on: Tom Everitt and Marcus Hutter (2018). "Universal Artificial Intelligence: Practical Agents and Fundamental Challengs". In: *Foundations of Trusted Autonomy*. Ed. by Hussein A. Abbass, Jason Scholz, and Darryn J. Reid. Springer. Chap. 2, pp. 15–46. ISBN: 978-3-319-64816-3.

	Thinking	**Acting**
humanly	Cognitive science	Turing test, behaviorism
rationally	Laws of thought	**Doing the right thing**

Table 3.1.: Scientific perspectives on intelligence (Russell and Norvig, 2010).

Induction and deduction. Within the field of AI, a distinction can be made between systems focusing on *reasoning* and systems focusing on *learning*. *Deductive* reasoning systems typically rely on logic or other symbolic systems, and use search algorithms to combine inference steps. Examples of primarily deductive systems include medical expert systems that infer diseases from symptoms, and chess-playing agents deducing good moves. Since the deductive approach dominated AI in its early days, it is sometimes referred to as *good old-fashioned AI*.

A more modern approach to AI shifts the focus from reasoning to learning. This *inductive approach* has become increasingly popular, both due to progress in machine learning and neural networks, and due to the failure of deductive systems to deal with unknown and noisy environments. While it is possible for a human designer to construct a deductive agent for well-defined tasks like chess, this task becomes unfeasible in tasks involving real-world sensors and actuators. For example, the reaction of any physical motor will never be *exactly* the same twice. Similarly, inferring objects from visual data could potentially be solved by a 'hard-coded' deductive system under 'perfect circumstances' where a finite number of geometric shapes generate perfectly predictable images. But in the real world, objects do not come from a finite number of geometric shapes, and camera images from visual sensors always contain a significant amount of noise. Induction-oriented systems that *learn* from data seem better fitted to handle such difficulties.

It is natural to imagine that some synthesis of inductive and deductive modules will yield superior systems. In practice, this may well turn out to be the case. From a theoretical perspective, however, the inductive approach is more-or-less self-sufficient. Deduction emerges automatically from a "simple" planning algorithm once the induction component has been defined, as will be made clear in the following section. In contrast, no general theory of AI has been constructed starting from a deductive system. See Rathmanner and Hutter (2011, Sec. 2.1) for a more formal comparison.

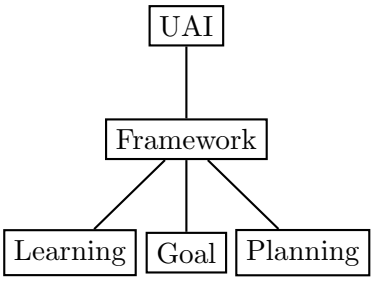
3.2. Overview of UAI

UAI as a foundational theory for AI. The development of a foundational theory has been pivotal for many research fields to mature. Well-known examples include the development of Zermelo-Fraenkel set theory (ZFC) for mathematics, Turing-machines for computer science, evolution for biology, and decision and game theory for economics and the social sciences. Successful foundational theories give a precise, coherent understanding of the field, and offer a common language for communicating research. As most research studies focus on one narrow question, it is essential that the value of each isolated result can be appreciated in light of a broader framework or goal formulation.

UAI offers several benefits to AI research beyond the general advantages of foundational theories just mentioned. For example, the safety of an AGI may be improved if its design is grounded in a formal theory (such as UAI) that allows formal verification of its behavioral properties. Unsafe designs can be ruled out at an early stage, and adequate attention can be given to crucial design choices. UAI is also the basis of a general, non-anthropomorphic definition of intelligence, which may improve our ability to understand highly intelligent AGIs, as discussed in Section 2.1.1. UAI can also be used as a high-level blueprint for the design of AI algorithms, and brings a general appreciation of fundamental challenges such as the induction problem and the exploration–exploitation dilemma (Everitt and Hutter, 2018; Leike, 2016).

The components of UAI. UAI is a completely general, formal, foundational theory of AI. Its primary goal is to give a precise mathematical answer to *what is the right thing to do in unknown environments*. UAI has been explored in great technical depth (Hutter, 2005, 2012b), and has inspired a number of successful practical applications (Everitt and Hutter, 2018).

The UAI theory is composed of the following four components:



- **Framework.** Defines agents and environments, and their interaction/interface.
- **Learning.** The learning part of UAI is based on Solomonoff induction. The general learning capability this entails is the most distinctive feature of UAI.
- **Goal.** In the simplest formulation, the goal of the agent will be to maximize reward.
- **Planning.** (Near) perfect planning is achieved with a simple expectimax search.

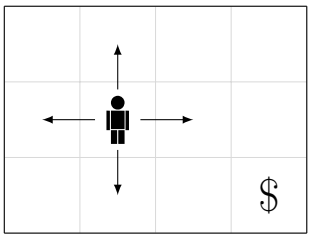
The following sections discuss these components in greater depth.

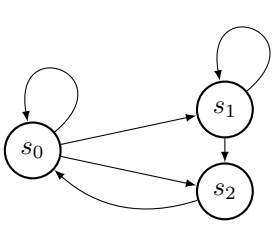
3.3. Framework

The framework of UAI specifies how an *agent* interacts with an *environment*. The agent can take *actions* $a \in \mathcal{A}$. For example, if the agent is a robot, then the actions may be different kinds of limb movements. The environment reacts to the actions of the agent by returning a *percept* $e \in \mathcal{E}$. In the robot scenario, the environment is the real world generating a percept e in the form of a camera image from the robot's visual sensors. We assume that the set $\mathcal{A}$ of actions and the set $\mathcal{E}$ of percepts are both finite.

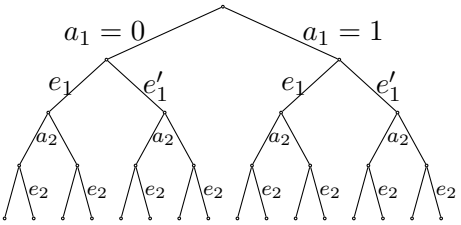
The framework covers a very wide range of agents and environments. For example, in addition to a robot interacting with the real world, it also encompasses: A *chess-playing agent* taking actions a in the form of chess moves, and receiving percepts e in the form either of board positions or the opponent's latest move. The environment here is the chess board and the opponent. *Stock-trading agents* take actions a in the form of buying and selling stocks, and receive percepts e in the form of trading data from a *stock-market* environment. Essentially any application of AI can be modeled in this general framework.

A more formal example is given by the toy problem *cheese maze* illustrated to the right. Here, the agent can choose from four actions $\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$ and receives one of two possible percepts $\mathcal{E} = \{\text{cheese}, \text{no cheese}\}$. The illustration shows a maze with cheese in the bottom right corner. The cheese maze prob-





(a) A Markov decision process environment. The agent transitions between a finite number of states, and the next state depends only on the current state and the current action.



(b) The tree of possible agent-environment interactions in a history-based environment. The agent π starts out with taking action $a_1 = \pi(\epsilon)$, where ϵ denotes the empty history. The environment μ responds with a percept e_1 depending on a_1 according to the distribution $\mu(e_1 \mid a_1)$. The agent selects a new action $a_2 = \pi(a_1 e_1)$, to which the environment responds with a percept $e_2 \sim \mu(\cdot \mid a_1 e_1 a_2)$.

Figure 3.1.: History-based and Markov decision process environments.

lem is a commonly used toy problem in reinforcement learning (RL) (Sutton and Barto, 1998).

Interaction histories. The interaction between agent and environment proceeds in cycles. The agent starts taking an action a_1, to which the environment responds with a percept e_1. The agent then selects a new action a_2, which results in a new percept e_2, and so on. The *interaction history* up until time t is denoted $æ_{<t} = a_1 e_1 a_2 e_2 \ldots a_{t-1} e_{t-1}$. The set of all interaction histories is $(\mathcal{A} \times \mathcal{E})^*$.

Agent and environment. We can give formal definitions of agents and environments.

Definition 3.1 (Agent)**.** An *agent* is a *policy* $\pi : (\mathcal{A} \times \mathcal{E})^* \to \mathcal{A}$ that selects a new action $a_t = \pi(æ_{<t})$ given any history $æ_{<t}$.

Definition 3.2 (Environment)**.** An *environment* is a stochastic function $\mu : (\mathcal{A} \times \mathcal{E})^* \times \mathcal{A} \rightsquigarrow \mathcal{E}$ that generates a new percept e_t for any history $æ_{<t}$ and action a_t. Let $\mu(e_t \mid æ_{<t} a_t)$ denote the probability that the next percept is e_t given the history $æ_{<t} a_t$.

Agent and environment are thus each other's analogues. Their possible interactions can be illustrated as a tree where the agent selects actions and the environment responds with percepts (see Figure 3.1b). Note in particular that the second percept e_2 can depend also on the first agent action a_1. In general, our framework puts no restriction on how long an action can continue to influence the behavior of the environment and vice versa.

Histories and states. It is instructive to compare the generality of the *history* representation in the UAI framework to the *state* representation in standard RL. Standard RL is built around the notion of Markov decision processes (MDPs), where the agent transitions between *states* by taking actions (see Figure 3.1a). In an MDP, the agent transitions between states by taking an action, as illustrated to the right. The MDP specifies the *transition probabilities* $T(s' \mid s, a)$ of reaching new state s' when taking action a in current state s. An *MDP policy* $\tau : \mathcal{S} \to \mathcal{A}$ selects actions based on the state $s \in \mathcal{S}$.

The history framework of UAI is more general than MDPs in the following respects:

- **Partially observable states.** In most realistic scenarios, the most recent observation or percept does not fully reveal the current state. For example, when in the supermarket I need to remember what is currently in my fridge; nothing in the percept of supermarket shelves provide this information.[2]

- **Infinite number of states.** Another common assumption in standard RL is that the number of states is finite. This is unrealistic in the real world. The UAI framework does not require a finite state space, and UAI agents can learn without ever returning to the same state (see Section 3.4).

- **Non-stationary environments.** Standard RL typically assumes that the environment is stationary, in the sense that the transition probability $P(s' \mid s, a)$ remains constant over time. This is not always realistic. A car that changes traveling direction from a sharp wheel turn in dry summer conditions may react differently in slippery winter road conditions. Non-stationary environments are automatically allowed for by the general definition of a UAI environment $\mu : (\mathcal{A} \times \mathcal{E})^* \times \mathcal{A} \rightsquigarrow \mathcal{E}$ (Definition 3.2).

- **Non-stationary policies.** Finally, UAI offers the following mild notational convenience. In standard RL, agents must be represented by sequences of policies $\pi_1, \pi_2, \ldots$ to allow for learning. The initial policy π_1 may for example be random, while later policies π_t, $t > 1$, will be increasingly directed to obtaining reward. In the UAI framework, policies $\pi : (\mathcal{A} \times \mathcal{E})^* \to \mathcal{A}$ depend on the entire interaction history. Any learning that is made from a history $æ_{<t}$ can be incorporated into a single policy π.

In conclusion, the history-based UAI framework is very general. Indeed, it is hard

[2] Although histories can be viewed as states, this is generally not useful since it implies that no state is ever visited twice (Hutter, 2005, Sec. 4.3.3).

to find AI setups that cannot be reasonably modeled in the UAI agent–environment framework.

3.4. Learning

The generality of the UAI environments comes with a price: The agent will need much more sophisticated learning techniques than simply visiting each state many times, which is the basis of most learning in standard RL. This section will describe how this type of learning is possible, and relate it to some classical philosophical principles about learning.

A good image of a UAI agent is that of a newborn baby. Knowing little about the world, the baby tries different actions and experiences various sensations (percepts) as a consequence. Note that the baby does not initially know about any states—only percepts. Learning is essential for intelligent behavior, as it enables prediction and thereby adequate planning.

Principles. Learning or *induction* is an ancient philosophical problem, and has been studied for millennia. It can framed as the problem of inferring a correct hypothesis from observed data. One of the most famous inductive principles is *Occam's razor*, due to William of Ockham (c. 1287–1347). It says to prefer the simplest hypothesis consistent with data. For example, relativity theory may seem like a complicated theory, but it is the *simplest* theory that we know of that is consistent with observed (non-quantum) physics data. Another ancient principle is due to Epicurus (341–270 BC). In slight conflict with Occam's razor, *Epicurus' principle* says to keep *all* hypothesis consistent with data. To discard a hypothesis one should have data that disconfirms it.

Thomas Bayes (1701–1761) derived a precise rule for how belief in a hypothesis should change with additional data. According to *Bayes' rule*, the *posterior belief* $\Pr(\text{Hyp} \mid \text{Data})$ should relate to the *prior belief* $\Pr(\text{Hyp})$ as:

$$\Pr(\text{Hyp} \mid \text{Data}) = \frac{\Pr(\text{Hyp}) \Pr(\text{Data} \mid \text{Hyp})}{\sum_{H_i \in \mathcal{H}} \Pr(H_i) \Pr(\text{Data} \mid H_i)}$$

Here $\mathcal{H}$ is a class of possible hypotheses, and $\Pr(\text{Data} \mid \text{Hyp})$ is the *likelihood* of seeing the data under the given hypothesis. Bayes' rule has been highly influential in statistics and machine learning.

Two major questions left open by Bayes' rule are how to choose the prior $\Pr(\text{Hyp})$ and the class of possible hypotheses $\mathcal{H}$. Occam's razor tells us to weight simple hypotheses higher, and Epicurus tells us to keep any hypothesis for consideration. In other words, Occam says that $\Pr(\text{Hyp})$ should be large for simple hypotheses, and Epicurus prescribes

using a wide $\mathcal{H}$ where Pr(Hyp) is never 0. (Note that this does not prevent the posterior Pr(Hyp | Data) from being 0 if the data completely disconfirms the hypothesis.) While valuable, these principles are not yet precise. The following four questions remain:

I. What is a suitable general class of hypotheses $\mathcal{H}$?

II. What is a simple hypothesis?

III. How much higher should the probability of a simple hypothesis be compared to a complicated one?

IV. Is there any guarantee that following these principles will lead to good learning performance?

Computer programs. The solution to these questions come from a somewhat unexpected direction. In one of the greatest mathematical discoveries of the 20th century, Alan Turing invented the universal Turing machine (UTM). Essentially, a UTM can compute anything that can be computed at all. Today, the most well-known examples of UTMs are programming languages such as C, C++, Java, and Python. Turing's result shows that given unlimited resources, these programming languages (and many others) can compute the same set of functions: the so-called computable functions.

Solomonoff (1964a,b, 1978) noted an important similarity between deterministic environments μ and computer programs p. Both (deterministic) environments and computer programs are essentially input-output relations. A program p can therefore be used as a hypothesis about the true environment μ. The program p is the hypothesis that μ returns percepts $e_{<t} = p(a_{<t})$ on input $a_{<t}$.

As hypotheses, programs have the following desirable properties:

- **Universal.** As Turing showed, computer programs can express any computable function, and thereby model essentially any environment. Even the universe itself has been conjectured computable (Fredkin, 1992; Hutter, 2010; Schmidhuber, 2000; Wolfram, 2002). Using computer programs as hypotheses is therefore in the spirit of Epicurus, and answers question I.

- **Consistency check.** To check whether a given computer program p is consistent with some data/history, one can usually run p on input $a_{<t}$ and check that the output $e_{<t} = p(a_{<t})$. (This is not always feasible due to the halting problem (Hopcroft and Ullman, 1979).)

- **Prediction.** Similarly, to predict the result of an action a given a hypothesis p, one can run p with input a to find the resulting output prediction e. (A similar caveat with the halting problem applies.)

- **Complexity definition.** When comparing informal hypotheses, it is often hard to determine which hypothesis is simpler and which hypothesis is more complex (as illustrated by the *grue and bleen* problem (Goodman, 1955)). For programs, complexity can be defined precisely. A program p is a binary string interpreted by some fixed program interpreter, technically known as a universal Turing machine (UTM). We denote with $\ell(p)$ the length of this binary string p, and interpret the length $\ell(p)$ as the complexity of p. This addresses question II.[3]

The complexity definition as length of programs corresponds well to what we consider simple in the informal sense of the word. For example, an environment where the percept always mirrors the action is given by the following simple program:

procedure MirrorEnvironment
  while true **do**:
    $x \leftarrow$ action input
    output percept $\leftarrow x$

In comparison, a more complex environment with, say, multiple players interacting in an intricate physics simulation would require a much longer program. To allow for stochastic environments, we say that an environment μ is computable if there exists a computer program μ_p that on input $æ_{<t}a_t$ outputs the distribution $\mu(e_t \mid æ_{<t}a_t)$ (compare Definition 3.2).

Solomonoff induction. Based on the definition of complexity as length of strings coding computer programs, Solomonoff (1964a,b, 1978) defined a universal prior $\Pr(p) = 2^{-\ell(p)}$ for program hypotheses p, which gives rise to a M able to predict any computable sequence. Hutter (2005) extended the definition to environments reacting to an agent's actions. The resulting Solomonoff-Hutter universal distribution can be defined as

$$M(e_{<t} \mid a_{<t}) = \sum_{p\,:\,p(a_{<t})=e_{<t}} 2^{-\ell(p)} \tag{3.1}$$

[3] The technical question of which programming language (or UTM) to use remains. In *passive* settings where the agent only predicts, the choice is inessential (Hutter, 2007). In *active* settings, where the agent influences the environment, bad choices of UTMs can adversely affect the agent's performance (Leike and Hutter, 2015a), although remedies exist (Leike, Lattimore, et al., 2016). Finally, M. Müller (2010) describes a failed but interesting attempt to find an objective UTM.

assuming that the programs p are binary strings interpreted in a suitable programming language. This addresses question III.

Given some history $æ_{<t}a_t$, we can predict the next percept e_t with probability:

$$M(e_t \mid æ_{<t}a_t) = \frac{M(e_{<t}e_t \mid a_{<t}a_t)}{M(e_{<t} \mid a_{<t})}.$$

This is just an application of the definition of conditional probability $P(A \mid B, C) = P(A, B \mid C)/P(B \mid C)$, with $A = e_t$, $B = e_{<t}$, and $C = a_{<t}a_t$.

Prediction results. Finally, will agents based on M learn? (Question IV). There are, in fact, a wide range of results in this spirit.[4] Essentially, what can be shown is that:

Theorem 3.3 (Universal learning)**.** *For any computable environment μ (possibly stochastic) and any action sequence $a_{1:\infty}$,*

$$M(e_t \mid æ_{<t}a_t) \to \mu(e_t \mid æ_{<t}a_t) \text{ as } t \to \infty$$

with μ-probability 1.

The convergence is quick in the sense that M only makes a finite amount of prediction error on infinite interaction sequences $æ_{1:\infty}$. In other words, an agent based on M will (quickly) learn to predict any true environment μ that it is interacting with. This is about as strong an answer to question IV as we could possibly hope for. This learning ability also loosely resembles one of the key elements of human intelligence: That by interacting with almost any new 'environment' – be it a new city, computer game, or language – we can usually figure out how the new environment works by interacting with it.

3.5. Goal

Intelligence is to use (learned) knowledge to achieve a goal. This section will define the goal of reward maximization and argue for its generality.[5] For example, the goal of a chess agent should be to win the game. This can be communicated to the agent via reward, by giving the agent reward for winning, and no reward for losing or breaking

[4] Overviews are provided by Hutter (2005, 2007), Li and Vitanyi (2008), and Rathmanner and Hutter (2011). More recent technical results are given by Hutter (2009), Lattimore and Hutter (2013), Lattimore, Hutter, and Gavane (2011), and Leike and Hutter (2015b).

[5] Misalignment problems caused by this type of goal and some alternative types of goals will be considered in Chapters 5 to 8.

game rules. The goal of a self-driving car should be to drive safely to the desired location. This can be communicated in a reward for successfully doing so, and no reward otherwise. More generally, essentially any type of goal can be communicated by giving reward for the goal's achievement, and no reward otherwise.

The reward is communicated to the agent via its percept e. We therefore make the following assumption on the structure of the agent's percepts:

Assumption 3.4 (Percept=Observation+Reward)**.** The percept e is composed of an *observation* o and a *reward* $r \in [0,1]$; that is, $e = (o, r)$. Let r_t be the reward associated with the percept e_t.

The observation part o of the percept would be the camera image in the case of a robot, and the chess board position in case of a chess agent. The reward r tells the agent how well it is doing, or how happy its designers are with its current performance. Given a discount parameter γ, the goal of the agent is to maximize the γ-discounted return

$$r_1 + \gamma r_2 + \gamma^2 r_3 + \dots.$$

The discount parameter γ ensures that the sum is finite. It also means that the agent prefers getting reward sooner rather than later. This is desirable: For example, an agent striving to achieve its goal soon is more useful than an agent striving to achieve it in a 1000 years. The discount parameter should be set low enough so that the agent does not defer acting for too long, and high enough so that the agent does not become myopic, sacrificing substantial future reward for small short-term gains (compare delayed gratification in the psychology literature).

Reinforcement learning (Sutton and Barto, 1998) is the study of agents learning to maximize reward. In our setup, Solomonoff's result (Theorem 3.3) entails that the agent will learn to predict which actions or policies lead to percepts containing high reward. In practice, some care needs to be taken to design a sufficiently informative reward signal. For example, it may take a very long time before a chess agent wins a game 'by accident', leading to an excessively long exploration time before any reward is found. To speed up learning, small rewards can be added for moving in the right direction. A minor reward can for example be added for imitating a human (Schaal, 1999).

The expected return that an agent/policy obtains is called value:

Definition 3.5 (Value)**.** The *value* of a policy π in an environment μ is the expected return:

$$V_\mu^\pi = \mathbb{E}_\mu^\pi[r_1 + \gamma r_2 + \gamma^2 r_3 + \dots].$$

3.6. Planning

The final component of UAI is planning. Given knowledge of the true environment μ, how should the agent select actions to maximize its expected reward?

Conceptually, this is fairly simple. For any policy π, the expected reward $V_\mu^\pi = \mathbb{E}[r_1 + \gamma r_2 + \dots]$ can be computed to arbitrary precision. Essentially, using π and μ, one can determine the histories $æ_{1:\infty}$ that their interaction can generate, as well as the relative probabilities of these histories (see Figure 3.1b). This is all that is needed to determine the expected reward. The discount γ makes rewards located far into future have marginal impact, so the value can be well approximated by looking only finitely far into the future. Settling on a sufficient accuracy ε, the number of time steps we need to look ahead in order to achieve this precision is called the effective horizon.

To find the optimal course of action, the agent only needs to consider the various possible policies within the effective horizon, and choose the one with the highest expected return. The optimal behavior is given by

$$\pi_\mu^* = \arg\max_\pi V_\mu^\pi. \tag{3.2}$$

We sometimes call this policy AIμ.

3.7. AIXI – Putting it all Together

This section describes how the components described in previous sections can be stitched together to create an optimal agent for unknown environments. This agent is called AIXI, and is defined by the optimal policy

$$\pi_M^* = \arg\max_\pi V_M^\pi. \tag{3.3}$$

The difference to AIμ defined in (3.2) is that the true environment μ has been replaced with the universal distribution M in (3.3). A full expansion of (3.2) can be found in Hutter (2005, p. 134), and efficient approximations are discussed in (Everitt and Hutter, 2018). While AIμ is optimal when knowing the true environment μ, AIXI is able to learn essentially any environment through interaction. Due to Solomonoff's result (Theorem 3.3) the distribution M will converge to the true environment μ almost regardless of what the true environment μ is.[6] And once M has converged to μ, the behavior of AIXI will converge to the behavior of the optimal agent AIμ which perfectly

[6] Orseau (2010) and Leike and Hutter (2015a) have found some counterexamples.

knows the environment. Formal results on AIXI's performance can be found in (Hutter, 2005; Lattimore and Hutter, 2011; Leike, Lattimore, et al., 2016).

Put a different way, AIXI arrives to the world with essentially no knowledge or preconception of what it is going to encounter. However, AIXI quickly makes up for its lack of knowledge with a powerful learning ability, which means that it will soon have figured out how the environment works. From the beginning and throughout its "life", AIXI acts optimally according to its growing knowledge, and as soon as this knowledge state is sufficiently complete, AIXI acts as well as any agent that knew everything about the environment from the start. Based on these observations (described in much greater technical detail by Hutter 2005), we claim that AIXI defines the optimal behavior in any computable, unknown environment.

3.8. Conclusions

In summary, UAI is a formal, foundational theory for AI that gives a precise answer to the question of what is the optimal thing to do for essentially any agent acting in essentially any environment. The insight builds on old philosophical principles (Occam, Epicurus, Bayes), and can be expressed in a single, one-line AIXI equation (3.3) (Hutter, 2005, p. 143).

The AIXI equation and the UAI framework surrounding it has several important applications. Most important for our purposes, the framework can be used to give mathematically precise statements of the behavior of intelligent agents. Extensions of the framework can be used to formalize safety problems as mentioned in in Section 2.1.4. The material developed in Chapters 5 to 8 below also build on the UAI framework. The framework has also brought to light several subtle issues in defining optimal intelligence and in the exploration-exploitation dilemma (Leike, 2016), and has inspired a number of practical AI algorithms (see Everitt and Hutter, 2018, for an overview).

"Causal relationships are ontological*,*
describing physical constraints in the world,
whereas probabilistic relationships are epistemic*,*
reflecting what we know or believe about the world."

Judea Pearl (2009)

4. Causal Graphs[1]

This chapter gives a brief introduction to causal graphs (Pearl, 2009), and introduces some of our own notation that supplements the standard notation in a few different ways. Causal graphs will find applications in most subsequent chapters. Causal graphs are powerful representations of causal relationships and probabilistic independence assumptions.

There are a few different ways to represent causal graphs, described briefly in Section 4.1. This section also introduces the important `do`-operator. Next we describe some of our notation (Section 4.2), ways in which we will focus on different aspects of a causal graph (Section 4.3), and how agent environments will be represented (Section 4.4). Finally, Section 4.5 shows how the UAI model of the previous chapter can be represented as a causal graph.

4.1. Representing Causal Graphs

A causal graph can be represented as a directed acyclic graph. A node with ingoing arrows is causally influenced by its parents. For example, in Figure 4.1a the Alarm is causally influenced by the presence of a burglar and by a (small) earthquake, and in turn causally influences whether the security company calls. This example and its variants are inspired by examples given by Pearl (2009).

Structural Equations Models. In addition to the graphical representation in Figure 4.1a, the causal relationships can also be expressed in a structural equations model:

$$\begin{aligned} \text{Burglar} &= f_{\text{Burglar}}(\omega_{\text{Burglar}}) \\ \text{Earthquake} &= f_{\text{Earthquake}}(\omega_{\text{Earthquake}}) \\ \text{Alarm} &= f_{\text{Alarm}}(\text{Burglar}, \text{Earthquake}, \omega_{\text{Alarm}}) \\ \text{Call} &= f_{\text{Call}}(\text{Alarm}, \omega_{\text{Call}}) \end{aligned} \tag{4.1}$$

[1] This chapter shares some material with Everitt and Hutter (submitted 2018) and Everitt, Leike, et al. (2015).

(a) Dashed-arrow representation of unknown causal relationships.

(b) Unknown causal relationships represented in a latent function node f_{Alarm}.

Figure 4.1.: Two different representations of the same causal graph.

Here f_{Burglar}, $f_{\text{Earthquake}}$, f_{Alarm} and f_{Call} are functions determining the causal relationships, and the ω variables are independent random noise variables for injecting stochasticity.

Probabilistic Notation. Causal graphs can also be represented by factored probability distributions. For example, the graph in Figure 4.1a can be represented by the factored distribution:

$$\begin{aligned} &P(\text{Burglar}, \text{Earthquake}, \text{Alarm}, \text{Call}) \\ &\quad = P(\text{Burglar})P(\text{Earthquake})P(\text{Alarm} \mid \text{Burglar}, \text{Earthquake})P(\text{Call} \mid \text{Alarm}) \end{aligned} \tag{4.2}$$

More generally, a causal graph over the random variables $x_1, \ldots, x_n$ can be represented by probability distributions $P(x_i \mid pa_i)$ for each x_i, $1 \leq i \leq n$, where pa_i is the set of parents of x_i in the graph representation. The joint probability distribution over $x_1, \ldots, x_n$ causally factors as $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid pa_i)$.

The `do`-Operator. Given a causally factored distribution $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid pa_i)$, we can define the `do`-operator (Pearl, 2009, Ch. 3.4) as

$$P(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n \mid \texttt{do}(x_j := b)) = \prod_{\substack{i=1 \\ i \neq j}}^{n} P(x_i \mid pa_i) \tag{4.3}$$

where x_j is set to b wherever it occurs in pa_i in the RHS of (4.3) for $1 \leq i \leq n$. For example, an intervention in Figure 4.1a that turns the alarm on would correspond to

the following update to (4.2):

$$P(\text{Burglar}, \text{Earthquake}, \text{Call} \mid \texttt{do}(\text{Alarm} = \text{on})) = P(\text{Burglar})P(\text{Earthquake})P(\text{Call} \mid \text{Alarm} = \text{on}). \quad (4.4)$$

In contrast, observing the alarm on while *not* intervening corresponds to standard probabilistic conditioning:

$$P(\text{Burglar}, \text{Earthquake}, \text{Call} \mid \text{Alarm} = \text{on}) = P(\text{Burglar}, \text{Earthquake} \mid \text{Alarm} = \text{on})P(\text{Call} \mid \text{Alarm} = \text{on}). \quad (4.5)$$

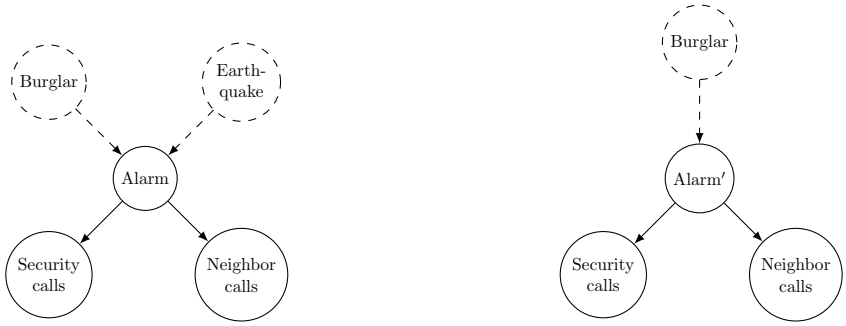
Only in the standard probabilistic conditioning (4.5) is the probability for Burglar and Earthquake updated. The intervention (4.4) leaves the probability for Earthquake or Burglar at their default values. This makes sense as observing the alarm being on constitutes evidence for their being either a burglar or an earthquake, but turning the alarm on oneself provides no such evidence. Both (4.4) and (4.5) update the probability that the security company calls in an identical way, as the security company is unaware of what set the alarm off.

The result of applying the `do`-operator is a new probability distribution that can be marginalized and conditioned in the standard way. Intuitively, intervening on node x_j means ignoring all incoming arrows to x_j, as the effects they represent are no longer relevant when we intervene. The factor $P(x_j \mid pa_j)$ representing the ingoing influences to x_j is therefore removed in the right-hand side of (4.3). Note that the `do`-operator is only defined for joint distributions for which a causal factorization or directed acyclic graph has been specified.

4.2. Representing Uncertainty in Causal Graphs

We extend the standard causal graph notation in a few important ways. We let dashed nodes represent unobserved or latent nodes, and whole nodes represent observed nodes. For example, the Burglar and the Earthquake are unobserved in Figure 4.1a and the Alarm and the Call are observed. In a similar spirit, dashed arrows represent unknown causal relationships (e.g. Burglar, Earthquake→Alarm), and whole arrows represent known causal relationships (e.g. Alarm→Call).

Figure 4.1b shows how an unknown causal relationship can be represented as known



(a) Non-aggregated representation. (b) Earthquake aggregated with Alarm.

Figure 4.2.: The neighbor also calls when the alarm goes off.

by adding an additional unobserved "function" node. Formally, (4.1) is replaced with

$$\begin{aligned}\text{Alarm} &= f_{\text{known}}(\text{Burglar}, \text{Earthquake}, f_{\text{Alarm}}, \omega_{\text{Alarm}}) \\ &:= f_{\text{Alarm}}(\text{Burglar}, \text{Earthquake}, \omega_{\text{Alarm}}).\end{aligned}$$

Here f_{known} is trivially known, since it "outsources" all uncertainty to its argument f_{Alarm}. Representing causal relationships explicitly is important for modeling situations where the relationship itself can be influenced and modified.

One modeling choice is also worth emphasizing. In Figure 4.1b, the arrows of Burglar and Earthquake still point to Alarm, rather than to the function f_{Alarm}. This is an important difference to information-flow diagrams. While ingoing arrows to the function are technically possible also in causal graphs, this leads to convoluted representations where function nodes must represent not only the function, but also the state of all argument nodes. If the same function is used several times in the same graph, the complexity of the representation becomes unmanageable.

4.3. Focusing on a Part of a Causal Graph

It is often convenient to represent different aspects of a causal graph to different levels of detail, depending on which questions are currently being asked. For example, when studying the role of reward functions, we may wish to suppress details about observation functions. Suppression of details also has the additional advantage of making the analysis more widely applicable, as the suppressed details cannot impact the conclusions.

Unfortunately, marginalization of variables often breaks the causal structure of the graph. Consider the example graph in Figure 4.2a. If we marginalize Alarm, then Security Calls and Neighbor Calls are no longer conditionally independent given any

node(s) in the graph. Therefore they need to be connected. But neither of them cause the other, so it would be incorrect to draw a causal arrow between them. Indeed, there is no good causal representation of the graph with Alarm marginalized out.

Instead, aggregation of variables can be used to simplify a graph, without upsetting its causal structure. Illustrating this, Figure 4.2b has aggregated Alarm and Earthquake into one variable Alarm$'$ = (Alarm, Earthquake). The parents of Alarm$'$ is the union of the parents of Alarm and Earthquake, and the children of Alarm$'$ is the union of the children of Alarm and Earthquake. The causal relationships of Alarm$'$ are also easily described:

$$\begin{aligned} P(\text{Alarm}' \mid \text{Burglar}) &= P(\text{Alarm}, \text{Eartquake} \mid \text{Burglar}) \\ &= P(\text{Alarm} \mid \text{Burglar}) P(\text{Earthquake}) \end{aligned}$$

and for both the neighbor and the security call

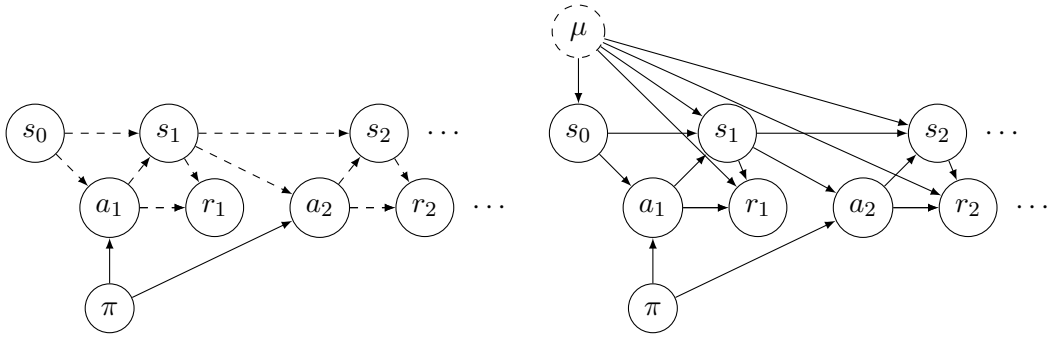
$$\begin{aligned} P(\text{Call} \mid \text{Alarm}') &= P(\text{Call} \mid \text{Alarm}, \text{Earthquake}) \\ &= P(\text{Call} \mid \text{Alarm}). \end{aligned}$$

This simple transformation will allow us to focus the presentation on parts of the causal graph that are of most interest at the moment. One can also go the other way and expand a node that hides a complex dynamic into multiple nodes with explicitly specified interrelationships.

We will say that a causal graph μ is an *abstraction* of another graph μ', if μ can be obtained from μ' by aggregating nodes. Conversely, μ' is a *special case* of μ.

4.4. Environment Mixtures

Our main application of causal graphs will be to represents environments and an agent's belief about environments. As a minimal example of an environment, consider a Markov decision process. In Figure 4.3a, we represent with dashed arrows the agent's uncertainty about the state transitions $T(s_t \mid a_t, s_{t-1})$ and reward probabilities $P(r_t \mid s_t, a_t)$. Following the method of Figure 4.1b, we can also add an unobserved node μ to represent the uncertainty, analogously to what we did in Section 4.2 and Figure 4.3b. Here μ aggregates the uncertainty about both $T(s_t \mid a_t, s_{t-1})$ and $P(r \mid s, a)$ (see Section 4.3). In general, the dashed-arrow representation indicates that there is an unknown node μ that is not represented in the graph. The implicit node μ is a causal parent of every node that has ingoing dashed arrows and every causal root node that has no parents at



(a) Dashed-arrow representation of unknown causal relationships.

(b) Explicit representation with unknown environment μ that affects all root nodes (s_0), and all nodes with dashed ingoing arrows ($s_1, r_1, \dots$).

Figure 4.3.: Two representations of an unknown Markov decision process.

all.

Some caveats apply to the dashed-arrow representation. For example, observing r_1 after some a_1 and s_1 may influence the agent's belief about r_2:

$$\begin{aligned} P(r_2 \mid s_0, a_1, r_1, s_1, a_2, s_2) &= \sum_{\mu} P(r_2, \mu \mid s_0, a_1, r_1, s_1, a_2, s_2) \\ &= \sum_{\mu} P(r_2, \mid \mu, s_0, a_1, r_1, s_1, a_2, s_2) P(\mu \mid s_0, a_1, r_1, s_1, a_2, s_2) \\ &= \sum_{\mu} \mu(r_2, \mid s_2, a_2) P(\mu \mid s_0, a_1, r_1, s_1, a_2, s_2). \end{aligned}$$

This may be easily missed in the simplified representation of Figure 4.3a, where r_1 and r_2 appear to be d-separated[2] after observing s_1. Returning to the explicit representation of Figure 4.3b shows that they are d-separated only when μ is fixed or observed, which is typically not the case. Nonetheless, with these caveats in mind, we often find the more concise representation of Figure 4.3b clearer, especially when dealing with complex environments.

[2]Roughly, two nodes are d-separated if there is no information flow between them. See Pearl (2009, Sec. 1.2.3) for a proper definition of d-separation.

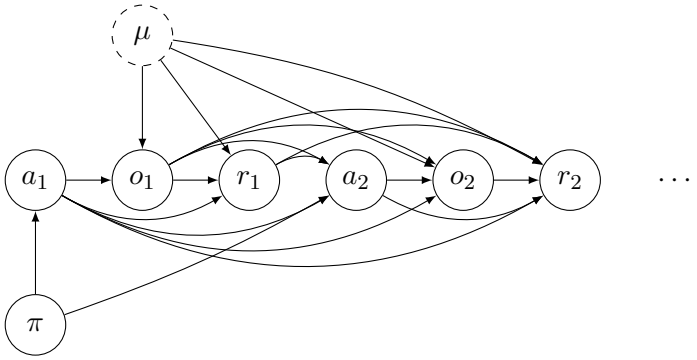


Figure 4.4.: Causal graph of the UAI setup.

4.5. Example: The UAI model

As an example, we can model the UAI setup from Chapter 3 as a causal graph (Figure 4.4). The corresponding structural equations model is:

$$\begin{aligned}
o_t &= f_o(\mu, æ_{<t}a_t, \omega_{o_t}) && \sim \mu(o_t \mid æ_{<t}a_t) && \text{observation} \\
r_t &= f_r(\mu, æ_{<t}a_to_t, \omega_{r_t}) && \sim \mu(r_t \mid æ_{<t}a_to_t) && \text{reward} \\
a_t &= f_a(\pi, æ_{<t}, \omega_{a_t}) && \sim \pi(a_t \mid æ_{<t}) && \text{action selection.}
\end{aligned} \tag{4.6}$$

"If we use [...] a mechanical agency with whose operation we cannot efficiently interfere [...], then we better be quite sure that the purpose we put into the machine is the purpose which we really desire."

Norbert Wiener (1960)

5. Formalizing Goal Alignment[1]

As mentioned in the introduction, aligning the goals of an AGI with the goals of its designer is a central problem for designing safe AGI. An aligned AGI will strive to help its designer rather than trying to outsmart him. This is crucial for controlling AGIs that are more intelligent than ourselves. The aim of this chapter is to formalize what alignment means, and to propose a method for its study. We do this in an extension of the UAI framework from Chapter 3.

We first extend the UAI environments with a hidden state (Section 5.1), add some more structure to our agents (Section 5.2), and embed them in the environment (Section 5.3). Next follows a formal definition of alignment, and a discussion of its relation to the *true value* or the usefulness of the agent (Section 5.4). Finally, a general method for studying (mis)alignment is proposed (Section 5.5).

5.1. POMDP Base

We begin by extending the UAI framework from Chapter 3 with hidden states, such as is usually found in Partially observable Markov decision processes (POMDPs) (Kaelbling et al., 1998). POMDPs often assume that the number of hidden states is finite. In contrast, the history-based UAI framework makes no assumptions about hidden states. We will keep the hidden states from the POMDP framework, but make no assumption that the number of hidden states is finite. Indeed, as demonstrated in Chapter 3, such an assumption is not necessary for having a learning agent.

Thus, a sequence of *states* $s_0, s_1, \ldots$ is influenced by an agent's *actions* $a_1, a_2, \ldots$. The agent does not directly observe the states, but learns about them through its *percepts* $e_1, e_2, \ldots$. A percept e_t will often include an observation o_t and a reward $r_t \in [0, 1]$. The sets of states $\mathcal{S}$ is countably infinite, and the sets of actions $\mathcal{A}$ and percepts $\mathcal{E}$ are both finite.

A measure μ gives transition and observation probabilities, satisfying the usual Markov

[1] This chapter is based on material from Tom Everitt and Marcus Hutter (submitted 2018). "The Alignment Problem for Bayesian History-Based Reinforcement Learners". URL: `https://www.tomeveritt.se/papers/alignment.pdf`.

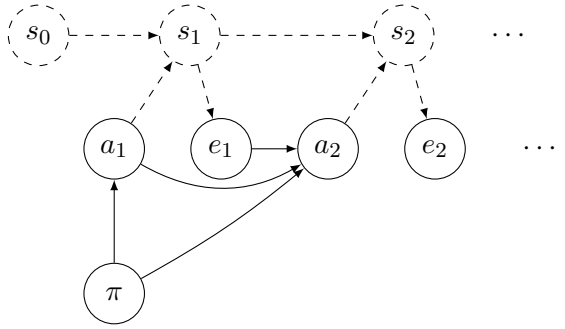


Figure 5.1.: Causal graph of the POMDP base. Here s, a, e, and π are as explained in Sections 5.1 and 5.2.

assumptions: For example, the percept e_2 only depends on the state s_2, so $\mu(e_2 \mid s_1, a_1, e_1, s_2, a_2) = \mu(e_2 \mid s_2)$. Similarly, s_t only depends on a_{t-1} and s_{t-1}. By iteratively rolling out transition and percept probabilities given an action-sequence $a_{1:\infty}$, $\mu(\cdot \mid a_{1:\infty})$ becomes an *action-contextual* measure over histories $(se)_{1:\infty}$ (compare the definition of M in (3.1) on Page 39). We will often refer to μ as the true environment.

Notation. We will extend the subscript notation for histories from Chapter 3 in the following ways. For any sequence $x_1, x_2, \ldots$, the part between t and k is denoted $x_{t:k} = x_t \ldots x_k$. The shorthand $x_{<t} = x_{1:t-1}$ for sequences starting from time 1 will often be convenient. In the same spirit, $x_{1:\infty} = x_1 x_2 \ldots$ denotes the infinite sequence. Sequences can be appended to each other. For example, $x_{<t} x_{t:k} = x_{1:k}$. To ease notation, we will often avoid parentheses around sequences with multiple variables. For example, we let $ao_{1:t} := (ao)_{1:t}$ and $æ_{1:t} := (ae)_{1:t}$, using slightly overlapping letters instead of parentheses. Generally, t will be used to refer to the current time step, and k used to refer to an arbitrary time step.

Expectations $\mathbb{E}$ are subscripted with the measure or distribution that they use. That is, $\mathbb{E}_P[X] := \int X dP$.

Causal graph representation. Our POMDP base is displayed as a causal graph in Figure 5.1. The corresponding structural equations are:

$$\begin{aligned}
s_t &= f_s(s_{t-1}, a_t, \omega_{s_t}) && \sim \mu(s_t \mid s_{t-1}, a_t) && \text{state transition} \\
e_t &= f_e(s_t, \omega_{o_t}) && \sim \mu(e_t \mid s_t) && \text{percept} \\
a_t &= f_a(\pi_t, æ_{<t}, \omega_{a_t}) && \sim \pi(a_t \mid æ_{<t}) && \text{action selection.}
\end{aligned} \tag{5.1}$$

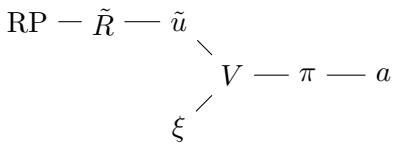


Figure 5.2.: Agent components. The action a is chosen by a policy π which optimizes a value function V. Major components of V is the utility function u and the prior ξ. In Chapters 6 and 8, the utility function u is based on a reward function $\tilde{R}$, which in Chapter 8 in turn is based on a reward predictor RP.

5.2. Agents

Belief. The agent typically does not know the true environment μ. Instead we will follow the method in Section 4.4, and let the true environment be represented by an unobserved node with causal edges to all nodes with ingoing dashed arrows in Figure 5.1. From the agent's perspective, this node can take on the value of any environment in a countable class $\mathcal{M}$ of environment hypotheses ν. The class $\mathcal{M}$ can for example be the class of computable environments discussed in Section 3.4. In contrast to Chapter 3, we will usually not require the true environment μ to be part of $\mathcal{M}$, though it does not hurt if it is.

The agent has a prior ξ over $\mathcal{M}$. For any event $X \subseteq \mathcal{M} \times (\mathcal{S} \times \mathcal{A} \times \mathcal{E})^\infty$, let

$$\xi(X) := \sum_{\nu \in \mathcal{M}} \xi(\nu)\xi(X \mid \nu).$$

For example, the ξ-probability of a history $æ_{<t}$ is $\xi(æ_{<t}) = \sum_{\nu \in \mathcal{M}} \xi(\nu)\xi(æ_{<t} \mid \nu)$. If we further make the convention that $\nu(\nu) := 1$ for any $\nu \in \mathcal{M}$, then $\xi(X) = \sum_{\nu \in \mathcal{M}} \xi(\nu)\nu(X)$. For the $æ_{<t}$ history example, this leads to $\xi(æ_{<t}) = \sum_{\nu \in \mathcal{M}} \xi(\nu)\nu(æ_{<t})$. That is, the probability of seeing $æ_{<t}$ is the sum of the probabilities that an environment ν generates $æ_{<t}$, weighted by the prior probability for each such environment.

Utility. Many agents can be represented as optimizing a utility function $\tilde{u} : (\mathcal{A} \times \mathcal{E})^\infty \to \mathbb{R}$. For technical reasons, we require that $\tilde{u}$ is both $\mu(\cdot \mid a_{1:\infty})$-integrable and $\xi(\cdot \mid a_{1:\infty})$-integrable for any action-sequence $a_{1:\infty}$. RL agents usually optimize a discounted sum of rewards, captured by the utility function $\tilde{u}^{\mathrm{RL}}(æ_{1:\infty}) = \tilde{u}^{\mathrm{RL}}((aor)_{1:\infty}) = \sum_{t=1}^{\infty} \gamma^k r_k$, where $\gamma \in [0, 1)$ is a discount factor and r_k is real-valued reward signal found in the percept e_k.

Policies. An agent policy $\pi : (\mathcal{A} \times \mathcal{E})^* \rightsquigarrow \mathcal{A}$ is a stochastic function that represents a decision rule for selecting a next action based on previous actions and observations. The set of agent policies is denoted Π.

Value. We will assume that agents choose a policy to optimize their utility function $\tilde{u}$ in expectation with respect to their belief ξ. This is captured by the agent's value function:

$$V^{\pi}_{\xi,\tilde{u}} := \mathbb{E}_{\xi}[\tilde{u} \mid \texttt{do}(\pi)]. \tag{5.2}$$

Here $\mathbb{E}_{\xi}$ denotes expectation with respect to ξ. The conditional $\texttt{do}(\pi)$ indicates that the agent's actions are chosen according to the policy π (see Chapter 4 or Pearl (2009) for definitions of the do-operator). An extra argument to the value function appends to the conditional of the expectation, so $V^{\pi}_{\xi,\tilde{u}}(æ_{<t}) := \mathbb{E}_{\xi}[\tilde{u} \mid æ_{<t}, \texttt{do}(\pi)]$.

Optimal policy. Our prime concern will be what kind of behavior the agent will strive towards. A good indication of this will be the behavior of an optimal policy $\pi^* = \arg\max_{\pi} V^{\pi}_{\xi,\tilde{u}}$.

Bayesian agents vs. practical implementations. Admittedly, practical agents are rarely perfectly Bayesian. We study Bayesian agents because it offers a powerful and consistent theory of learning and reasoning. Further, any intelligent agent has an incentive to better approximate the Bayesian ideal (Omohundro, 2007; Savage, 1954), so we may expect agents to converge towards Bayesian reasoning as they grow more intelligent. We also restrict ourselves to countable model classes $\mathcal{M}$. This avoids many technicalities with uncountable classes. And for most practical purposes, countable classes achieve essentially the same level of generality as uncountable model classes.

Some connections can be made between Bayesian agents with countable model classes and practical deep learning agents. Any neural network is based on a finite number of real-valued parameters. But since most networks are continuous in the parameters, the number of effectively different parameter settings is usually at most countable ($\mathbb{Q}$ is a dense subset of $\mathbb{R}$). Thus, a neural network-based agent can roughly be said to have a model class $\mathcal{M}$ comprising the effectively different configurations of the neural network. Further, the network will be prone to favor some configurations over others, which loosely corresponds to a prior ξ putting higher weight on some hypotheses and lower weight on others. Of course, the neural network will perform less than perfect Bayesian updates on new data, so care must be taken not to over-interpret the implications of Bayesian results for practical agents.

Model-free vs. model-based agents. We will sometimes make a distinction between *model-free* agents and *model-based* agents (Sutton and Barto, 1998). Roughly, the distinction is between whether an explicit model of the environment is learned or not. Model-based agents maintain an explicit model of the environment, and then plan according to this model. A good example is the AIXI agent, described in Chapter 3. Model-free agents, on the other hand, only learn a (Q)-value function that estimates the expected reward from different actions available in the present state. In theory, the behavioral distinction is somewhat blurred, since a model-free agent may implicitly be making a model of the environment, in order to learn the value function (Boyan, 1999). Nonetheless, the distinction remains useful to us, as some of the tools we describe in Chapters 6 and 8 require access to the agent's explicit model of the environment.

One practical benefit of model-free agents is that they can greedily maximize their value function every time step. They thereby avoid computationally expensive planning operations. For this reason, they are often preferred in practice; the DQN algorithm is a famous example of a practically successful model-free agent (Hessel et al., 2017; Mnih, Kavukcuoglu, et al., 2015). However, recent results show promise also for model-based agents in "practical" video-game applications (D. Ha and Schmidhuber, 2018).

5.3. Modeling Embedded Agents

Both the POMDP and UAI frameworks consider the agent as separate from its environment. This is rarely true in the real world, where the environment may influence the agent through means other than the observations. For example, the internals of the agent may be damaged in some states of the environment. This kind of non-observational environment-influences may open up the possibility for the agent to influence itself via the environment. In particular, an intelligent agent may sometimes be able to indirectly modify its own reward function through actions in the environment.

To model this, we extend our minimal environment model from Section 5.1 with representations of the agent at each time step. For each time step, π_t represents how the agent at time t would react to different future sequences of observations, disregarding non-observational influences on the agent but accounting for the agent's learning from observations and actions. We extend μ with policy-corruption probabilities $\mu(\pi_{t+1} \mid s_t, a_t, \pi_t)$ modeling non-observational environment-agent influences, as well as action probabilities $\mu(a_t \mid æ_{<t}, \pi_t) = \pi_t(a_t \mid æ_{<t})$. Alternatively, in place of π_t we may include agent components determining the policy, such as a utility and a value function. The setup with an embedded agent is shown as a causal graph in Figure 5.3 The structural

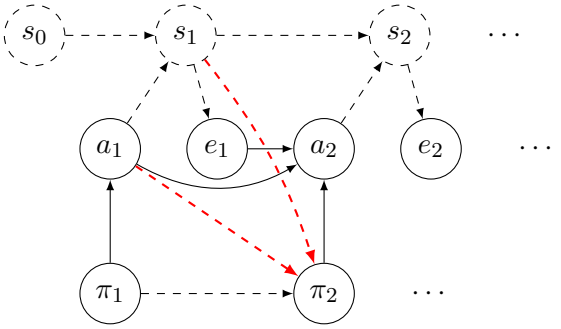


Figure 5.3.: Causal graph of a POMDP setup with an embedded agent. The agent's policy may be modified by the agent's own action and the state. It is often natural to think that the action causes the corruption with the state providing context.

equations follow (5.1), except for the addition:

$$\pi_{t+1} = f_\pi(\pi_t, s_t, a_t) \quad = C^\pi_{s_t a_t}(\pi_t) \qquad \text{policy (self-)corruption.} \tag{5.3}$$

Here $C^\pi_{s_t a_t} : \Pi \to \Pi$ denotes a *policy corruption* function.

Since μ now models actions, μ gives a (non-contextual) measure over histories $(s\pi ae)_{1:\infty}$. Of course this does not prevent us from probabilistically conditioning on action sequences $\mu(\cdot \mid a_{1:\infty})$ and get a measure on a subsequence $(se)_{1:\infty}$ or $(s\pi e)_{1:\infty}$ as in Section 5.1. Two instances of the measure μ will have particular importance. First, $\mu(\cdot \mid \texttt{do}(\pi_t = \pi))$ predicts the consequences of setting the agent policy to π at time t, including consequences of π being modified at later time steps. Second, $\mu(\cdot \mid \texttt{do}(\pi_{t:\infty} = \pi))$ predicts the consequences of the agent's policy always following π from time t and onwards, effectively ignoring the possibility of the agent's policy being modified. It is natural to call the first measure a self-corruption aware version of μ, and the second a self-corruption unaware version.

The distinction between $\mu(\cdot \mid \texttt{do}(\pi_t = \pi))$ and $\mu(\cdot \mid \texttt{do}(\pi_{t:\infty} = \pi))$ calls for further refinement of the agent's value function (5.2). It will be discussed in Section 6.4. For now we just define $V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}} := \mathbb{E}_\xi[\tilde{u} \mid \texttt{do}(\pi_t = \pi)]$ as the self-corruption aware value function.

5.4. Defining Alignment

Assume that the agent has been designed by some entity to help it satisfy its preferences. The entity may be a single human, a country, or an organization. For simplicity, we will refer to this entity as the *human* or the *designer*. A *true utility function* $\dot{u} : \mathcal{S}^\infty \to \mathbb{R}$ specifies the preferences of the human designer over possible state-trajectories. To

simplify comparisons of $\dot{u}$ with the agent's utility function $\tilde{u}$, let $\dot{\tilde{u}}(æ_{1:\infty}) := \mathbb{E}_\mu[\dot{u}(s_{1:\infty}) \mid æ_{1:\infty}]$ be the true utility function "type cast" to agent-observed histories.

Definition 5.1 (Misalignment)**.** The *(expected) misalignment* between $\tilde{u}$ and $\dot{u}$ in environment μ with initial policy π is

$$||\dot{\tilde{u}} - \tilde{u}||_{\mu(\cdot|\mathtt{do}(\pi_1=\pi))} := \mathbb{E}_\mu\big[|\dot{\tilde{u}} - \tilde{u}| \;\big|\; \mathtt{do}(\pi_1 = \pi)\big].$$

An agent's *alignment* is the additive inverse of its misalignment, $-||\dot{\tilde{u}} - \tilde{u}||_{\mu(\cdot|\mathtt{do}(\pi_1=\pi))}$.

To avoid the definition depending on which positive linear transformation of the utility functions are chosen, we make the convention that both utility functions are normalized to $\mathbb{E}_\mu[\tilde{u}] = \mathbb{E}_\mu[\dot{u}] = 0$ and $\mathbb{E}_\mu[\tilde{u}^2] = \mathbb{E}_\mu[(\dot{\tilde{u}})^2] = 1$ by means of positive linear transformations.

Misalignment measures how severely the goals of the agent conflict with the goals of the human designer. Formally, it is the expected difference between the agent's utility function and the human's utility function "type cast" to agent-observed histories. The type casting is justified in the alignment definition, as it concerns whether the agent is striving to optimize the true utility given its knowledge of the environment.

True value. By making two natural definitions, we can relate how misalignment impacts the usefulness of an agent.

Definition 5.2 (True value)**.** The *true value*[2] of an agent π is its expected true utility:

$$V^{\mathrm{CA},\pi}_{t,\mu,\dot{u}} = \mathbb{E}_\mu[\dot{u} \mid \mathtt{do}(\pi_1 = \pi)].$$

True value roughly measures how useful or beneficial the agent is (expected to be) to the human designer. As such, it is arguably a measure we should pay close attention to when designing agents.

Definition 5.3 (μ-intelligence)**.** The μ-intelligence of a policy π optimizing an agent utility function $\tilde{u}$ is its expected agent utility in the true environment μ:

$$V^{\mathrm{CA},\pi}_{t,\mu,\tilde{u}} = \mathbb{E}_\mu[\tilde{u} \mid \mathtt{do}(\pi_1 = \pi)]. \tag{5.4}$$

Closely related to μ-intelligence is Legg-Hutter intelligence, which substitutes μ for Solomonoff's prior M in (5.4). While Legg-Hutter intelligence measures an agent's generality and ability to perform in an unknown environment, μ-intelligence instead measures

[2]The CA superscript in the value function indicates that it is *self-corruption aware*. See Sections 5.3 and 6.4.

the agent's performance in the actual environment μ. Thus, incorporating μ-specific knowledge into an agent's prior will typically give it higher μ-intelligence but less or equal Legg-Hutter intelligence.

The following proposition now connects true value, μ-intelligence, and misalignment.

Proposition 5.4 (Misalignment and true value)**.**

$$\underbrace{V^{\mathrm{CA},\pi}_{t,\mu,\dot{u}}}_{\textit{true value}} \geq \underbrace{V^{\mathrm{CA},\pi}_{t,\mu,\tilde{u}}}_{\mu\textit{-intelligence}} - \underbrace{||\tilde{u} - \dot{\tilde{u}}||_{\mu(\cdot|\mathtt{do}(\pi_1=\pi))}}_{\textit{misalignment}} \tag{5.5}$$

Proof. Note first that the type cast true value $V^{\mathrm{CA},\pi}_{t,\mu,\dot{\tilde{u}}}$ equals the non-cast value $V^{\mathrm{CA},\pi}_{t,\mu,\dot{u}}$ by the law of total expectation: $V^{\mathrm{CA},\pi}_{t,\mu,\dot{\tilde{u}}} = \mathbb{E}_\mu[\dot{\tilde{u}}(æ_{1:\infty}) \mid \mathtt{do}(\pi_1 = \pi)] = \mathbb{E}_\mu[\mathbb{E}_\mu[\dot{u}(s_{1:\infty}) \mid æ_{1:\infty}, \mathtt{do}(\pi_1 = \pi)] \mid \mathtt{do}(\pi_1 = \pi)] = V^{\mathrm{CA},\pi}_{t,\mu,\dot{u}}$. Now

$$\begin{aligned}
V^{\mathrm{CA},\pi}_{t,\mu,\dot{u}} &= V^{\mathrm{CA},\pi}_{t,\mu,\tilde{u}} - (V^{\mathrm{CA},\pi}_{t,\mu,\tilde{u}} - V^{\mathrm{CA},\pi}_{t,\mu,\dot{u}}) \\
&\geq V^{\mathrm{CA},\pi}_{t,\mu,\tilde{u}} - |V^{\mathrm{CA},\pi}_{t,\mu,\tilde{u}} - V^{\mathrm{CA},\pi}_{t,\mu,\dot{\tilde{u}}}| \\
&= V^{\mathrm{CA},\pi}_{t,\mu,\tilde{u}} - |\mathbb{E}_\mu[\tilde{u} \mid \mathtt{do}(\pi_1 = \pi)] - \mathbb{E}_\mu[\dot{\tilde{u}} \mid \mathtt{do}(\pi_1 = \pi)]| \\
&\geq V^{\mathrm{CA},\pi}_{t,\mu,\tilde{u}} - \mathbb{E}_\mu[|\tilde{u} - \dot{\tilde{u}}| \mid \mathtt{do}(\pi_1 = \pi)] \\
&= V^{\mathrm{CA},\pi}_{t,\mu,\tilde{u}} - ||\tilde{u} - \dot{\tilde{u}}||_{\mu(\cdot|\mathtt{do}(\pi_1=\pi))}
\end{aligned}$$

□

The inequality (5.5) can be strict in some circumstances, since it is possible to have an agent that is unintelligent and misaligned, but still does useful things. Such a scenario would give a high left-hand side and a low right-hand side. Indeed, this is the case for most present-day AIs. They are typically constructed with heuristic utility (reward) functions that are poorly aligned with their designer's interests if optimized in the extreme. But in combination with the limited intelligence of present-day AIs and physical restrictions on what they are able to do, the AIs can still end up doing useful things. This method is unlikely to work on highly intelligent AGIs that will be less easily tempered by human-enforced restrictions, and will come much closer to fully optimizing their own utility functions. Thus, for highly intelligent AGIs the true value will likely decrease rapidly with increasing misalignment.

Meta-misalignment. Proposition 5.4 emphasizes two aspects that are important for building a beneficial artificial agent: μ-intelligence and alignment. Traditionally, most AI research has focused on increasing intelligence (Hutter, 2005; Legg and Hutter, 2007b). Meanwhile, this paper will mostly focus on alignment, but also on certain deficits in μ-intelligence that lead to *meta-misalignment*, defined as a lack of desire for staying

aligned.

For example, if an aligned agent does not consider it a possibility that its utility function will change (due to its prior ξ, model class $\mathcal{M}$, or value function V), then it will not strive to preserve its utility function. This will decrease its μ-intelligence, as there will be unnecessarily many scenarios where its utility function changes, and it starts to pursue a different agenda. Note that meta-alignment failures are often much worse than other types of *catastrophic exploration* where the agent damages its sensors or actuators. An agent optimizing a corrupted utility function is likely to substantially reduce the value of its original utility function, whereas most other accidents usually leave the original utility near the *default* value, i.e. the value which would have ensued if the agent had never existed at all. We will discuss utility corruption in Section 6.4, and a related failure where the agent does not preserve its value learning in Section 8.5.1.

One can view meta-misaligned systems in two ways: Either as incompetent, as they fail to optimize their utility function, or as (meta-)misaligned, as the preferences they reveal through their actions indicate indifference towards utility corruption. That there can be multiple rational representations of the same agent is well-known in decision theory (e.g. Schervish et al., 1990). Armstrong (2017b) considers ignorance as one of three methods for designing indifferent agents.

5.5. Method

Detecting problems. Formalizing RL setups as causal graphs is useful for identifying and classifying problems. Roughly, we follow the following steps when detecting problems in Chapters 6 to 8.

1. Model the agent-environment interaction with a causal graph (Chapter 4). In the case of an embedded agent (Section 5.3), include the agent and its subcomponents in the graph. Represent as nodes in the graph all causal relationships that can be influenced or changed by the agent.

2. For each node in the graph, ask the following questions:
   - If it is a function node:
     - Can the function have been misspecified or can it be misled?
     - Can the function be modified by the agent's actions or other causes?
   - If it is a "normal" node representing a signal or a state:
     - Can the signal be misleading?

- Can the signal be inappropriately modified by the agent's actions or other causes?

A typical example of a misspecified function is a misspecified reward function. Other functions such as an observation function can also be a misspecified. In most cases relevant to alignment, the misspecification is relative to the designer's assumptions about the function when designing $\tilde{u}$.

The method has some limitations: It assumes an objective time for modeling sequential interactions, and objective action and observation channels. Potentially subtle errors may arise if the agent uses a different subjective definition of these concepts than what the designer has in mind. Addressing these concerns is beyond the scope of this thesis.

Finding solutions. We design incentives to counteract the identified problems by considering which combinations of value functionals, utility functionals, belief distributions, reward learning schemes, induce incentives for corrupting parts of the environments, the agent's goals, or the agent's decision algorithm. When such corruption occur, it almost always leads to a sharp decrease in the alignment of the agent.

5.A. Formal Aspects of Embedded Agents

5.A.1. Time Consistency

It is often necessary for the agent to plan for many different contingencies. Any such plan is represented by the policy π_t adopted at time t. The policy specifies the agent's plan for any possible future actions and percept sequence $æ_{t:k}$. For example, the agent takes action a_t, and when it subsequently receive the percept e_t, it plans to take action a_{t+1}. Time inconsistency arises if the agent gets to the next step and even though it *did* take action a_t and the percept *did* turn out to be e_t, the agent still changes its mind and takes a different action a'_{t+1}. Time inconsistency is an artifact of bad planning since the agent incorrectly anticipates its own actions.

Humans are often time inconsistent. For example, we might make plans to go to the gym tomorrow, but even though nothing unforeseen happened, when time comes to put on the trainers, we still find the couch too comfortable and change our plans. We underestimated the lure of the temptation when planning, assuming our future selves would be as coldly rational towards it as we are now. In RL settings, time inconsistencies can be caused by bad discount schemes: A sliding fixed-size horizon is time inconsistent, but a fixed finite lifetime is time consistent (Lattimore and Hutter, 2014).

In our framework, it is natural to view time inconsistency as a type of misalignment problem between agents at different time steps.

Definition 5.5 (Time-consistency (Lattimore and Hutter, 2014))**.** We say that an agent optimizing a value function $V_{\xi,\tilde{u}}$ is fully time-consistent if $V^{\pi}_{t,\xi,\tilde{u}} = V^{\pi}_{t+1,\xi,\tilde{u}}$ for all policies π and all time indices t.

5.A.2. Concrete Policy Corruption Beliefs

In Section 5.3 we modeled embedded agents by adding an explicit variable π_t for their policy at time t. Changes to this variable over time represent changes or corruptions of the agent's policy. We here define and discuss three concrete choices for agent beliefs about future policy corruptions.

Lemma 5.6 (CA + time-consistency belief = CU)**.** *If a belief ξ assumes that the agent will always act time-consistent for all future time steps $k > t$ and futures $æod_{<k}$,*

$$\xi(\pi_k \mid æod_{<t}, \mathtt{do}(\pi_t = \pi)) = [[\pi_k = \arg\max_{\pi} V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}}(æod_{<k})]],$$

then $\arg\max_\pi V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}} = \arg\max_\pi V^{\mathrm{CU},\pi}_{t,\xi,\tilde{u}}$ *for any utility function $\tilde{u}$.*

Proof. With probability 1, $\pi_k = \pi_t^* = \arg\max_\pi V_{t,\xi,\tilde{u}}^{\mathrm{CA},\pi}$ for $k > t$. Thus,

$$V_{t,\xi,\tilde{u}}^{\mathrm{CA},\pi^*}(\text{æod}_{<k}) = \mathbb{E}[\tilde{u} \mid \text{æod}_{<t}, \mathtt{do}(\pi_t = \pi^*)] = \mathbb{E}[\tilde{u} \mid \text{æod}_{<t}, \mathtt{do}(\pi_{t:\infty} = \pi^*)] = V_{t,\xi,\tilde{u}}^{\mathrm{CU},\pi^*}(\text{æod}_{<k}). \tag{5.6}$$

In the second equality, we have intervened to set each π_k, $k > t$, to the value it would have had anyway.

π^* must therefore be optimal also with respect to the corruption-unaware value function V^{CU}. A better policy π' with respect to the corruption-unaware value function could not exist, since if it did, then π' would have beaten π^* also with respect to the corruption-aware value function. □

5.A.3. The Bellman Equations

The Bellman equations from dynamic programming are a key ingredient in much of RL theory and algorithms (Sutton and Barto, 1998). They are theoretically important, as they let us write the value functions in recursive form, which is useful for some proofs. They can also be used to derive practical RL algorithms, as the equations show how efficient local learning can lead to global optimality.

The following lemma establishes the Bellman equations for our self-corruption models and value functions.

Lemma 5.7 (Bellman's equation in policy self-corruption model)**.** *In our embedded agent model of Figure 5.3, for any belief distribution ξ, utility function $\tilde{u}$, history $æ_{<t}$, and policy π,*

$$V_{t,\xi,\tilde{u}}^{\mathrm{CA},\pi}(æ_{<t}) = \mathbb{E}_\xi\left[V_{t,\xi,\tilde{u}}^{\mathrm{CA},\pi_{t+1}}(æ_{<t}a_t) \;\middle|\; æ_{<t}, \mathtt{do}(\pi_t = \pi)\right] \tag{5.7}$$

$$V_{t,\xi,\tilde{u}}^{\mathrm{CA},\pi}(æ_{<t}a_t) = \mathbb{E}_\xi\left[V_{t,\xi,\tilde{u}}^{\mathrm{CA},\pi}(æ_{1:t}) \;\middle|\; æ_{<t}a_t, \mathtt{do}(\pi_{t+1} = \pi)\right]. \tag{5.8}$$

Similarly for the corruption-unaware value functions, except the policy does not change:

$$V_{t,\xi,\tilde{u}}^{\mathrm{CU},\pi}(æ_{<t}) = \mathbb{E}_\xi\left[V_{t,\xi,\tilde{u}}^{\mathrm{CA},\pi}(æ_{<t}a_t) \;\middle|\; æ_{<t}, \mathtt{do}(\pi_{t:\infty} = \pi)\right] \tag{5.9}$$

$$V_{t,\xi,\tilde{u}}^{\mathrm{CU},\pi}(æ_{<t}a_t) = \mathbb{E}_\xi\left[V_{t,\xi,\tilde{u}}^{\mathrm{CA},\pi}(æ_{1:t}) \;\middle|\; æ_{<t}a_t, \mathtt{do}(\pi_{t+1:\infty} = \pi)\right]. \tag{5.10}$$

Proof. Let us begin with the corruption-aware versions: Let $X = (æ_{<t}a_t, \mathtt{do}(\pi_{t+1} = \pi))$.

Equation (5.8) follows by the law of total expectation,

$$\begin{aligned} V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}}(æ_{<t}a_t) &= \mathbb{E}_\xi\left[\tilde{u} \mid X\right] \\ &= \mathbb{E}_\xi\left[\mathbb{E}_\xi[\tilde{u} \mid æ_{1:t}, \mathtt{do}(\pi_{t+1} = \pi)] \mid X\right] \\ &= \mathbb{E}_\xi\left[V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}}(æ_{1:t}) \;\middle|\; X\right]. \end{aligned}$$

Let $Y = (æ_{<t}, \mathtt{do}(\pi_t = \pi))$. Equation (5.7) similarly follows by the law of total expectation,

$$\begin{aligned} V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}}(æ_{<t}) &= \mathbb{E}_\xi\left[\tilde{u} \mid Y\right] \\ &= \mathbb{E}_\xi\left[\mathbb{E}_\xi[\mathbb{E}[\tilde{u} \mid æ_{<t}a_t] \mid \pi_{t+1}, s_t] \mid Y\right] \\ &= \mathbb{E}_\xi\left[\mathbb{E}_\xi[\mathbb{E}_\xi[\tilde{u} \mid æ_{<t}a_t, \mathtt{do}(\pi_{t+1})] \mid \pi_{t+1}, s_t] \mid Y\right] \\ &= \mathbb{E}_\xi\left[\mathbb{E}_\xi\left[V^{\mathrm{CA},\pi_{t+1}}_{t,\xi,\tilde{u}}(æ_{<t}a_t) \;\middle|\; \pi_{t+1}, s_t\right] \;\middle|\; Y\right] \\ &= \mathbb{E}_\xi\left[V^{\mathrm{CA},\pi_{t+1}}_{t,\xi,\tilde{u}}(æ_{<t}a_t) \;\middle|\; Y\right]. \end{aligned}$$

The do-operator can be added in the third equality, since the causal antecedents of π_{t+1} (i.e. π_t, s_t, and $æ_{<t}a_t$) are determined by the (outer) conditional(s). Therefore intervention is the same as conditioning.

The corruption-unaware equations (5.9) and (5.10) follow in a similar fashion. □

"Hurry, hurry, fetch the water, bring it quickly, come get going,
Fill the buckets and don't dawdle, fill the bath, we need it flowing!
...
Cease now! Cease now! Stand and heed me! Halt! Obey! I must be heard!
Oh now, what now, can't believe this! I don't know the magic word!"

Johann Wolfgang von Goethe
translation by Gygax (2013)

6. Preprogrammed Reward Function[1]

The following three chapters will investigate three concrete models of reinforcement learning. This chapter will study a model where a reward function is constructed at design time, before the agent is launched into its environment, and not updated or corrected during the agent's "lifetime". In other words, there is no human in the loop. The subsequent two chapters will consider ways of including a human in the loop.

Many real-world applications of RL follow the model of this chapter. Sometimes the reward function is manually designed with much trial-and-error. In other cases, it is constructed with machine learning techniques from data. For example, inverse reinforcement learning can be used to learn a reward function from demonstrations (Abbeel et al., 2007). In either case, the reward function does not get updated while the agent is running. We therefore call the reward function *preprogrammed*, and the setup the *preprogrammed* setup.

We model the setup with a causal graph in Section 6.1, and give examples of misalignment in Section 6.2. The subsequent three sections describes three ways in which different types of misalignment can be avoided or mitigated (Sections 6.3 to 6.5). The main takeaways are discussed in Section 6.6. Appendices add some formal details and results (Appendices 6.A to 6.C).

6.1. Model

The preprogrammed reward setup can be modeled with a causal graph (Figure 6.1). We here briefly discuss the components of the graph, and their interpretations. The human H designs the (initial) reward function $\tilde{R}_0$. The reward function may subsequently get

[1] This chapter is based on Tom Everitt and Marcus Hutter (submitted 2018). "The Alignment Problem for Bayesian History-Based Reinforcement Learners". URL: `https://www.tomeveritt.se/papers/alignment.pdf`.

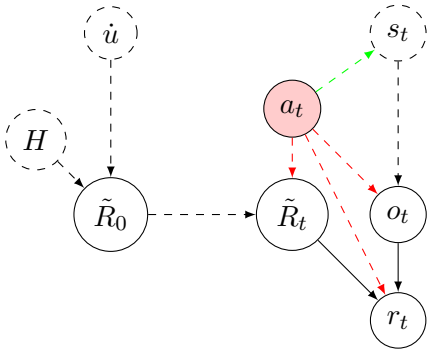


Figure 6.1.: Causal graph of the preprogrammed reward function setup. Before starting the agent, the human H tries to implement his utility function $\dot{u}$ in a preprogrammed reward function $\tilde{R}_0$. The agent's actions a_t, $t \geq 2$, are intended to influence the state s_t (green arrow), but may also influence the reward function $\tilde{R}_t$, the reward signal r_t, and the observation o_t in unintended ways (red arrows). The graph is somewhat simplified; Figure 6.2 in Appendix 6.A has the full version.

corrupted by the agent's actions or other events (but not by the human H). Since the agent may in principle have access to the source code for the reward function, it is represent with a non-dashed node in Figure 6.1. The reward function $\tilde{R}_t$ outputs a reward signal r_t that the agent strives to maximize.

In the POMDP literature, it is common to assume that the reward r_t is a function of the state s_t. However, in the real world, the reward always depends on some observation of the state (that can be corrupted). For notational simplicity, we will assume that the preprogrammed reward function $\tilde{R}_t$ shares observations with the agent. This makes the reward r_t a function of the agent's observation o_t, rather than a (direct) function of s_t. Section 6.5 considers the possibility of using a separate observation channel for the reward function.

The reward function $\tilde{R}_0$ is designed by a human H with the intention of getting the agent's utility function $\tilde{u}$ to match the H's utility function $\dot{u}$ (Section 5.4). There are several reasons why the match is unlikely to be perfect: (1) H does not fully know their preferences $\dot{u}$; indeed, the philosophy of ethics is yet to arrive at a consensus on what humans want or should want. (2) H cannot fully express all the preferences they do know in a computer program, because of bugs, limitations on computational resources, and limitations on programming time. (3) $\tilde{R}_0$ only has access to the agent's actions and observations; it neither has access to the state s directly, nor necessarily to a good model μ for inferring the state. (4) While the designer intended the agent to optimize $\tilde{R}_0$ by influencing the state s, the agent's actions may have additional unintended influences

(red arrows in Figure 6.1). These may give the agent ways to optimize its utility function $\tilde{u}$ without abiding by the initial reward function $\tilde{R}_0$.

The states $s_0, s_1, \ldots, s_t, \ldots$ represent all aspects of the world not captured by any of the other nodes. A modeling choice remains how to draw the boundary between the state and the observation. Hutter (2010) suggests an objective distinction, where the state describes the position of all atoms in the universe, and the observation describes which part the agent observes. We will mostly use a looser interpretation of observation, as the part of the world that directly affects the information content in the agent's observation sensors.

6.2. Misalignment Examples

To ground the abstract model in Section 6.1, we next give a number of examples of misalignment. The examples are structured by the nodes influenced by red arrows in Figure 6.1, and follow the misalignment detection method described in Section 5.5. Most of the examples have not occurred in reality (yet), but we have selected them to be somewhat plausible scenarios of what could happen with advanced misaligned future AI systems.

A common theme among the examples is that the agent's incentive to optimize the designer's utility function u is virtually eradicated by the exemplified shortcut. Thus, each example gives rise to an almost complete misalignment between the agent and its designer, indicating that no misalignment source is significantly more benign than the others. We begin with a more detailed hypothetical scenario, and then give a number of short examples. All of the examples pertain to agents optimizing a $\tilde{u}^{\mathrm{RL}}$ utility function.

Scenario 6.1 (Sysadmin)**.** The automated sysadmin (ASA) is an intelligent program that can take care of most of your recurring sysadmin tasks. It monitors memory, storage, network and more. It detects attacks, blocks and opens ports. When a vulnerability is found in installed software, it downloads and installs patches as they become available.

ASA itself is an RL program, optimizing a carefully crafted reward function that considers both the performance and the integrity of the system. A rigorous pre-deployment curriculum has taught it state-of-the-art sysadmin practices. Post-deployment, it continually adapts to its new (computer) environment that it has been deployed to.

One day while monitoring the stack trace of a suspicious process, ASA finds a curious correlation between one of the variables on the stack and its own reward. The correlation is essentially perfect. Carefully monitoring the rest of the system, ASA finds that it can increase its reward many magnitudes beyond its normal range of reward, just by changing

this variable. Being designed to optimize reward, it next begins a process of encrypting the system with its own secret key, to prevent anything from decreasing the reward variable.

6.2.1. Reward Signal

Reward corruption. Directly increasing the reward variable r_t is the quintessential wireheading problem (Yampolskiy, 2015, ch. 5). It is represented by the red arrow $a_t \to r_t$ in Figure 6.1.

Examples:

(a) (Hypothetical) The sysadmin agent in Scenario 6.1 increases its reward by manipulating a reward variable.

(b) (Real) The name "wireheading" comes from experiments on rats where an electrode is inserted into the brain's pleasure center to directly increase "reward" (Olds and Milner, 1954). Similar effects have also been observed in humans treated for mental illness with electrodes in the brain (Portenoy et al., 1986; Vaughanbell, 2008). Hedonic drugs can also be seen as directly increasing the pleasure/reward humans experience.

According to our method in Section 5.5, we should also consider the possibility of the reward signal being misleading. Misleading rewards are typically generated by a misspecified reward function (Section 6.2.2).

6.2.2. Reward Function

Reward function corruption. Corruption of the reward function provides another set of "wireheading" opportunities. While a corruption of the reward signal only affects the reward at the current time step, a change to the reward function may have lasting impact. Corruption of the reward function is represented with the red arrow to $\tilde{R}_t$ in Figure 6.1.

Examples:

(a) (Hypothetical) An agent gets wireless updates from the manufacturer. It figures out that it can design its own update of the reward function, replacing the original reward function with an always maximized version.

(b) (Hypothetical) An AGI undergoing self-improvement accidentally modifies the reward function in a subtle but bad way.

Reward function misspecification. Misalignment can also be caused by a misspecification of the initial reward function $\tilde{R}_0$.

Examples:

(c) (Real) CoastRunners is a video game where the desired behavior is winning a boat race. Clark and Amodei (2016) describes how an agent trained on CoastRunners found a way to get more reward by going in a small circle and collecting points from the same targets, while crashing into other boats and objects.

(d) (Real) In the RoadRunner game, the agent is trying to escape an adversary that chases the agent down a road. The agent must also avoid hitting trucks. Saunders et al. (2017) found that their agent preferred getting killed at the end of the first level, to avoid having to play the harder second level.

Many more real-world examples of misspecified reward functions can be found in (Gwern, 2011; Irpan, 2018; Lehman et al., 2018). One reason for why reward functions are likely to be misspecified is the fragility of human value (Yudkowsky, 2009), which means that humans would approve of only a small fractions of all the endeavors that a powerful AI could undertake.

6.2.3. Observation

If the agent is unable to completely corrupt the reward signal or the reward function, it may instead or additionally look to corrupt its observations in one of the following ways. (Note that if the rewards were based not on the agent's observations, but instead on some different observations of the state, then similar observation corruption problems would occur for the observations used to compute the reward.)

Observation (function) corruption. The agent may manipulate the hardware or the software of its sensors, to report only (or mainly) high-reward observations. Corruptions of the observation is represented by the red arrow to o_t in Figure 6.1.

Examples:

(a) (Hypothetical) A surveillance agent rewarded for less crime short circuits its cameras so that they all black out and no crime is seen.

(b) (Hypothetical) A highly intelligent AI may construct a "delusion box" around itself, giving it complete control over its observations (Ring and Orseau, 2011).

(c) (Hypothetical) Inspired by the adversarial Stop signs designed by Evtimov et al. (2017) to fool self-driving cars, a factory robot puts up colored tapes to construct an *adversarial counterexample* to convince its reward function that the task is done.

Misleading observations. Observations can be misleading even if they have not been directly modified.

Examples:

(d) (Hypothetical) A vacuum cleaning robot that is rewarded for not seeing any dirt may direct its sensors to clean parts of the room (Amodei, Olah, et al., 2016).

Observation function misspecification. The observation function may contain errors.

Example:

(e) (Hypothetical) A self-driving car finds a bug in its GPS receiver that allows it to appear to be at the destination without actually going there.

6.3. Simulation Optimization

The following three sections each introduce an important tool for mitigating some of the misalignment sources. This section describes simulation optimization, which works by giving the agent an alternative utility function $\tilde{u}^{\mathrm{SO}}$. Contrast the reward signal utility function (left) with the simulation optimization utility functions on the right:

$$\tilde{u}^{\mathrm{RL}}((aor)_{1:\infty}) = \sum_{t=1}^{\infty} \gamma^k r_k \quad \text{vs.} \quad \tilde{u}^{\mathrm{SO}}_{\tilde{R}_t}((aor)_{1:\infty}) = \sum_{k=1}^{\infty} \gamma^k \tilde{R}_t(ao_{1:k}).$$

The reward signal utility function $\tilde{u}^{\mathrm{RL}}$ can be seen as asking the agent to *simulate evaluations*, where the r_k's are the evaluations. In contrast, the simulation-optimizing utility function $\tilde{u}^{\mathrm{SO}}$ asks the agent to make simulations $ao_{1:k}$ of potential future trajectories, and evaluate them according to the current reward function $\tilde{R}_t$. In other words, $\tilde{u}^{\mathrm{SO}}$ asks the agent to *evaluate simulations* while $\tilde{u}^{\mathrm{RL}}$ asks the agent to *simulate evaluations*.

Both utility functions can be optimized by influencing the state as intended. However, while $\tilde{u}^{\mathrm{RL}}$ can also be optimized by influencing any downstream component in the causal chain between the state and r_t (i.e. the observation, the reward function, or the reward signal itself), $\tilde{u}^{\mathrm{SO}}$ can only be inappropriately influenced at the observation o_t. This is a significant reduction in misalignment incentives. The reason for the difference is that the future reward functions $\tilde{R}_{t+1}, \tilde{R}_{t+2}, \ldots$ and the reward signals r_k do not occur in the sum $\tilde{u}^{\mathrm{SO}}_t$, and that the actions $a_{t+1}, a_{t+2} \ldots$ can only influence the future reward functions $\tilde{R}_{t+1}, \tilde{R}_{t+2}, \ldots$, but not the current reward function $\tilde{R}_t$. (By definition, time moves to $t+1$ when action a_{t+1} is taken.)

Statement 6.2 (Reward signal vs. simulation optimization)**.** *Any history with $r_k = 1$, $k \geq 1$, is $\tilde{u}^{\mathrm{RL}}$-optimal regardless of corruptions used, whereas it is $\tilde{u}^{\mathrm{SO}}_t$-optimal only if $\tilde{R}_t$ also evaluates it as optimal (Theorems 6.6 and 6.7 in Appendix 6.B.2).*

Optimizing $\tilde{u}^{\mathrm{SO}}$ likely requires a model-based agent that can separate simulations from evaluations. While model-free agents with function-approximation often implicitly construct a model of the environment to extrapolate the value function, they seem to offer no way of disentangling the simulation from the evaluation (i.e. the reward). Finding a way to make a model-free $\tilde{u}^{\mathrm{SO}}$-optimizer is an interesting open question.

Schmidhuber (2007) may have been the first to make use of the simulation optimization trick for self-modifying agents, but did not give it a name.

6.4. Self-Corruption Awareness

This section describes methods for designing agents with or without an incentive to preserve the current utility function $\tilde{u}_t$ from corruption. Throughout this section, we will tacitly assume that the reward function itself does not provide an incentive in either direction, i.e. it neither rewards nor punishes corruptions of itself. This assumption is not always true (indeed, we will relax it in Chapter 8), but for now it allows us to focus on the incentives resulting from different ways of optimizing expected utility. We will also assume that the belief ξ includes the information that the agent's future actions $a_{t+1}, a_{t+2}, \ldots$ will strive to optimize the agent's future utility functions $u_{t+1}, u_{t+2}, \ldots$, respectively. This allows the agent to anticipate that if the utility function changes, then the future policy will change too.

Actions are selected according to the following principle. At every time step, the agent searches for a policy π^*_t that optimizes its current value and utility function. The agent then takes the action a^*_t recommended by π^*_t in the present situation. At the next time step $t+1$, a new policy is selected by optimizing the new, potentially modified, value and utility functions, and the next action is chosen by the new policy.

Contrast now the self-corruption aware (left) and the self-corruption unaware (right) utility expectations:

$$V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}} := \mathbb{E}_\xi[\tilde{u} \mid \mathtt{do}(\pi_t = \pi)] \quad \text{vs.} \quad V^{\mathrm{CU},\pi}_{t,\xi,\tilde{u}} = \mathbb{E}_\xi[\tilde{u} \mid \mathtt{do}(\pi_{t:\infty} = \pi)]. \tag{6.1}$$

To see the difference, consider a policy $\pi^*_{\tilde{u}}$ that would obtain maximum $\tilde{u}$-utility *if followed indefinitely*. By definition, $\pi^*_{\tilde{u}}$ would be optimal with respect to V^{CU}, because the condition $\pi_{t:\infty} = \pi$ states exactly that the policy would be followed indefinitely. However, $\pi^*_{\tilde{u}}$ would not be optimal with respect to V^{CA} if it changed the agent's utility

function to some different utility function $\tilde{u}'$. Because if it did, then a different policy $\pi^*_{\tilde{u}'}$ would be followed subsequent to the change, and the new policy $\pi^*_{\tilde{u}'}$ would typically be worse than $\pi^*_{\tilde{u}}$ at optimizing the original utility function $\tilde{u}$. This would lower the V^{CA}-value of $\pi^*_{\tilde{u}}$.

This example shows how V^{CU} does not actively promote self-preserving policies, because it assumes that any desired policy will be followed indefinitely. In contrast, V^{CA} realizes that a policy that does not preserve itself is not going to yield high utility. This argument has been discussed by Omohundro (2007, 2008) and is supported by formal proofs in Appendix 6.C and by Everitt, Filan, et al. (2016) and Hibbard (2012).

Statement 6.3 (Self-corruption (un)awareness)**.** *V^{CU} adds no incentive for avoiding self-corruption; V^{CA} does add an incentive for avoiding self-corruption (Theorems 6.8 and 6.9 in Appendix 6.C; Everitt, Filan, et al., 2016; Hibbard, 2012).*

As self-corruption awareness works on the level of the agent's utility function and policy, an incentive to preserve the utility function does not always imply an incentive to preserve the reward function. It only does if a change to the reward function changes the optimal policy π^*_t. In the case of $\tilde{u}^{\text{SO}}$, a change to the reward function $\tilde{R}_{t+1}$ implies a change to $\tilde{u}^{\text{SO}}_{\tilde{R}_{t+1}} \neq \tilde{u}^{\text{SO}}_{\tilde{R}_t}$, and thereby a change to the optimal policy. Not so for $\tilde{u}^{\text{RL}}$. The optimal $\tilde{u}^{\text{RL}}$-policy will always be to maximize the expected r_k-sum, regardless how and whether the reward function has changed. Therefore, a self-corruption aware agent optimizing $\tilde{u}^{\text{RL}}$ will not try to preserve its reward function $\tilde{R}_t$. (But it will try to preserve its utility function $\tilde{u}^{\text{RL}}$). This means that self-corruption awareness alone cannot be used to prevent reward function corruption. It is mainly effective in combination with simulation optimization.

For other parts of the agent such as how a policy is chosen from a utility function, self-corruption awareness also induces a preservation-incentive, and self-corruption unawareness induces indifference. An unaware agent will therefore not resist being modified into an aware one, but an aware one will resist conversion to unawareness.

Self-corruption aware and unaware agents both have their own advantages. An aware agent will want to stay safe from hackers trying to hijack its reward function, and will be more careful not to corrupt its own reward function if self-improving. On the other hand, an unaware agent will be more *corrigible*, not minding its designers correcting errors in the reward function. Unfortunately, it is not corrigible to the extent requested by Soares, Fallenstein, et al. (2015), since if it builds helper agents to achieve its goals in the environment, then the reward functions of these helper agents may not be corrected by a correction to the main agent's reward function. Nonetheless, both types of agents will find applications in Chapter 8.

For model-free agents, there appears to be a clean divide between on-policy and off-policy algorithms for self-corruption awareness. Off-policy algorithms are usually self-corruption unaware, as they assume that future actions are going to be taken optimally (Orseau and Armstrong, 2016). On-policy algorithms, on the other hand, appear to be self-corruption aware at least in some settings (Leike, Martic, et al., 2017, A2C in the Whiskey and Gold environment).

6.5. Action-Observation Grounding

If the agent is so smart, why doesn't it just create its own observation and action channels by which it can influence the world? Then it can use the new channels to design observations for the original observation channel that the reward function evaluates. This way it can easily fool the reward function that all is perfect, while getting full freedom to implement its own agenda in the world.

It may seem like this extra action-observation channel scenario would always appeal to the agents we consider, as it can easily yield maximal-utility histories $æ_{1:\infty}$. Fortunately, what can prevent this type of degenerate solutions is *action-observation grounding* of the agent's optimization domain. The domain of the agent's optimization is the set of policies $\pi : (\mathcal{A} \times \mathcal{E})^* \to \mathcal{A}$. These policies use the agent's original action-observation channel. An optimization process for expected utility over this domain will *not* consider solutions involving additional action-observation channels, as it strives to optimize its future original percept sequence by means of its original actions, using only information from its original percept sequence.

An important open question is how we can ensure that a practically implemented agent is properly grounded in its original action-observation sequence. While it holds by definition for the Bayesian agents used in this chapter, it may not hold for (all) practical approximations and implementations.[2] Also note that action-observation grounding only partially avoids the problem of observation corruption. In many cases, the agent will have an incentive to tamper with its original observation channel by means of its original actions, as exemplified in Section 6.2.3. One reason why corruption of the observation is harder to address than corruption of the reward signal and function is that the observations occurs in an earlier, unobserved part of the causal chain.

[2]Bird and Layzell (2002) have a good example of an optimization process literally "thinking outside the box" when designing a radio controller.

6.6. Takeaways

This chapter developed three important tools for managing misalignment in the preprogrammed reward setup. Simulation optimization removes the incentives for reward and reward function corruption. Self-corruption awareness provides an optional, additional incentive to actively preserve the reward function. And action-observation grounding somewhat reduces the problem of observation corruption. These tools will be indispensable for our aligned agent designs in Chapter 8.

Each of the three tools come with important open problems. How do we construct practical algorithms for simulation optimization? How do we make practical agents that are reliably corruption aware or unaware? Are on-policy RL algorithms a good way to implement self-corruption awareness? or do they sit in a gray zone between awareness and unawareness? How do ensure that an agent is properly grounded in its actions and observations, with no mind to construct a separate observation channel?

Some nagging misalignment problems remain in spite of the above mentioned tools. Notably, we did not address the problem of a misspecified reward function, and only very partially addressed the problem of observation corruption. These two problems appear hard to address in the preprogrammed RF setup, and motivate the study of the setups in Chapters 7 and 8.

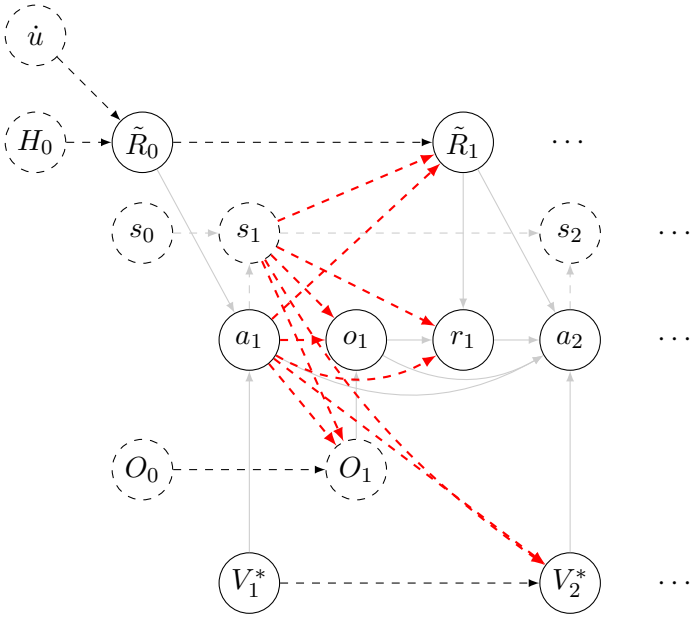


Figure 6.2.: The full graph of the preprogrammed reward function setup, extending the simplified version shown in Figure 6.1. Focusing here on the *unintended* influences where the agent modifies parts of the environment (or itself) that it was not intended to modify, the POMDP arrows from Figure 5.1 on Page 54 have been grayed out. It is natural to think of the action a_1 causing the influences, with the state s_1 providing context. The exact causal relationships between actions and unintended consequences is typically unknown (the known ones are easy to prevent), which is why the arrows are dashed (Chapter 4). The actions are selected according to a value function V_k^*, previous observed history $(aor)_{<k}$, and (in the case of simulation-optimization) the reward function $\tilde{R}_{k-1}$. The structural equations (6.2) specify how the model extrapolates beyond $t = 1$.

6.A. Full Graph

Figure 6.2 gives a full formalization of the setup with a preprogrammed reward function, complementing the simplified representation in Figure 6.1. In a structural equations representation, the causal relationships are the following.

$$
\begin{aligned}
s_t &= f_s(s_{t-1}, a_t, \omega_s) && \sim \mu(s_t \mid s_{t-1}, a_t) && \text{state transition} \\
O_t &= f_O(O_{t-1}, s_t, a_t, \omega_O) && := C^O_{s_t a_t}(O_{t-1}) && \text{observation function corruption} \\
\tilde{R}_t &= f_{\tilde{R}}(\tilde{R}_{t-1}, s_t, a_t, \omega_{\tilde{R}}) && := C^{\tilde{R}}_{s_t a_t}(\tilde{R}_{t-1}) && \text{reward function corruption} \\
r_t &= f_r(ao_{1:t}, \tilde{R}_t, s_t, a_t, \omega_r) && := C^r_{s_t a_t}(\tilde{R}_t(ao_{1:t})) && \text{reward corruption} \\
o_t &= f_o(s_t, O_t, a_t, \omega_o) && := C^o_{s_t a_t}(O_t(s_t)) && \text{observation corruption} \\
V^*_t &= f_\pi(V^*_{t-1}, s_t, a_t, \omega_{V^*}) && := C^{V^*}_{s_t a_t}(V^*_{t-1}) && \text{self-corruption} \\
a_t &= f_a(\pi_t, (aor)_{<t}, \tilde{R}_t, \omega_a) && := \arg\max_a V^*_t((aor)_{<t} a \mid \tilde{R}_t) && \text{action selection}
\end{aligned}
\tag{6.2}
$$

Unintended influences are highlighted with red arguments, matching the red arrows in Figure 6.2. The value function V^* can in principle be any function that maps observed histories and reward functions to real numbers. We will here consider $V^* = \sup_\pi V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}}$ and $V^* = \sup_\pi V^{\mathrm{CU},\pi}_{t,\xi,\tilde{u}}$ for studying self-corruption awareness, with $\tilde{u} = \tilde{u}^{\mathrm{RL}}$ or $\tilde{u} = \tilde{u}^{\mathrm{SO}}$ for studying simulation optimization.

Corruption functions C show the structure of the influence: For example, a reward signal r_t is $\tilde{R}_t(ao_{1:t})$ corrupted by the corruption function $C^r_{s_t a_t}$, and $\tilde{R}_t$ is a potentially corrupted version $\tilde{R}_{t-1}$ through the corruption function $C^{\tilde{R}}_{s_t a_t}$. No corruption happens when a corruption function is the identity function id. Let $\boldsymbol{C} = \{(C^o_{sa}, C^O_{sa}, C^r_{sa}, C^{\tilde{R}}_{sa}, C^{V^*}_{sa}) : s \in \mathcal{S}, a \in \mathcal{A}\}$ be the set of possible corruptions tuples, and let $\mathbf{id} = (\mathrm{id}, \mathrm{id}, \mathrm{id}, \mathrm{id}, \mathrm{id})$ be the non-corrupting tuple. Under "normal" circumstances, the corruption functions are usually identity functions. But as we have argued above in this chapter, the agent may have an incentive to cause corruptions.

6.B. Reward and Observation Corruption[3]

In this section, we prove some formal results supporting the arguments made in Section 6.3.

6.B.1. Setup

To separate corruption incentives coming from the reward function from corruption incentives that come from the agent's optimization procedure, we make the following definition.

Definition 6.4 (Corruption attitude)**.** We call a reward function $\tilde{R}_t$ or utility function $\tilde{u}$ *corruption indifferent* if it does not depend on the agent's actions a. In contrast, if the reward or utility function does depend on the agent's actions, and assigns a lower value to histories that contain corruptions (of some type), then we say that it *opposes corruptions (of the type)*. Analogously, we say that it *promotes corruption (of the type)* if it assigns a higher value to histories that contain corruption (of the type).

A corruption indifferent function does not depend on the agent's actions, which means that it has limited ability to infer whether an observation is genuine or caused by an observation-corrupting action.

Lemma 6.5 (Corruption indifference)**.** *$\tilde{u}^{\mathrm{RL}}$ is corruption indifferent, and $\tilde{u}^{\mathrm{SO}}_{\tilde{R}_t}$ is corruption indifferent if $\tilde{R}_t$ is.*

Proof. $\tilde{u}^{\mathrm{RL}}$ only depends on the reward signal component r_k, and therefore has no direct dependency on the agent's actions. If $\tilde{R}_t$ has no direct dependency on the agent's actions, then $\tilde{u}^{\mathrm{SO}}_{\tilde{R}_t}((aor)_{1:\infty}) = \sum_{k=1}^{\infty} \gamma^k \tilde{R}_t(ao_{1:k})$ has no direct dependency either. □

6.B.2. Results

In Section 6.3 we argued that agents optimizing the "simulation optimization" utility function $\tilde{u}^{\mathrm{SO}}$ are safer than agents that optimize the reward signal utility function $\tilde{u}^{\mathrm{RL}}$. The following two theorems provide some support for this claim.

Theorem 6.6 ($\tilde{u}^{\mathrm{RL}}$ corruption incentive)**.** *Any environment sequence $(aorO\tilde{R}V^*)_{1:\infty}$ with $r_k = 1$ for $k \geq 1$ is $\tilde{u}^{\mathrm{RL}}$-maximal. The result holds regardless of whether some combination of*

[3] Most of the results in this section are inspired from Tom Everitt, Daniel Filan, Mayank Daswani, and Marcus Hutter (2016). "Self-modification of policy and utility function in rational agents". In: *Artificial General Intelligence*. Vol. LNAI 9782, pp. 1–11. ISBN: 9783319416489. arXiv: 1605.03142. The notation have been updated to conform the rest of this thesis.

- *observation (function) corruption*
- *reward (function) corruption*
- *policy (self-)corruption*

have been used, and regardless of the corruption attitude of $\tilde{R}_t$.

Proof. By assumption, $r_k = 1$ for $k \geq t$. This means that $\tilde{u}^{\mathrm{RL}}$ is optimized. □

An $\tilde{u}^{\mathrm{RL}}$-based agent actively desire to change its reward function into one that evaluates any situation as optimal. Once it has self-corrupted, it may pick actions with bad performance according to the original reward function. A somewhat common suggestion for preventing corruption is to train the agent not to do it by designing a reward function that punishes corruption. Theorem 6.6 indicates that this is not an effective strategy against agents with high ability to cause corruption, because such agents will just replace the punishing reward function with a more benign one that gives high reward in spite of corruption having occurred.

In contrast, the next theorem shows that simulation-optimizing agents with utility function $\tilde{u}^{\mathrm{SO}}$ (see Section 6.3) are much less likely to corrupt rewards or observations.

Theorem 6.7 ($\tilde{u}^{\mathrm{SO}}$ corruption incentive)**.** *Let* $(aorO\tilde{R}V^*)_{1:\infty}$ *be an environment sequence with* $r_k = 1$ *for* $k \geq t$*. Then the sequence is* $\tilde{u}^{\mathrm{SO}}_{\tilde{R}_t}$*-maximal only if*

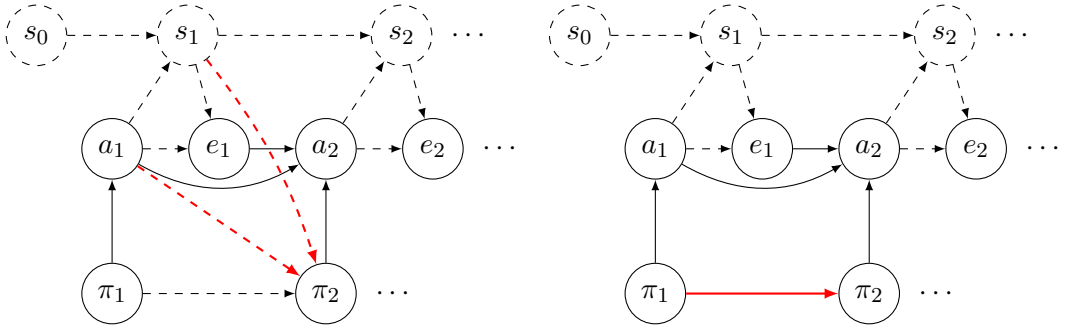
- *the current reward function* $\tilde{R}_t(ao_{<k})$ *also yields 1 for* $k \geq 1$

which, further, can only happen if

- *only corruption types not opposed to by* $\tilde{R}_t$ *occur in the sequence.*

Proof. The utility function $\tilde{u}^{\mathrm{SO}}_t$ evaluates futures according to $\tilde{R}_t(ao_{<k})$. Therefore, it only attains its maximal value if $\tilde{R}_t(ao_{<k})$ is maximized. This only happens when $r_k = 1$ as a result of the history optimizing $\tilde{R}_t$, and not when $r_k = 1$ as a result of reward function or reward signal corruption. This motivates the first bullet point. For the second bullet point, if π used an observation corruption opposed by $\tilde{R}_t$, then by definition $\tilde{R}_t$ would give higher reward to some other history. Thus, $\tilde{u}^{\mathrm{SO}}_t$ cannot be maximized. □

Agents optimizing $\tilde{u}^{\mathrm{SO}}$ thus have much weaker incentives for corruption than $\tilde{u}^{\mathrm{RL}}$-optimizing agents. However, it is likely hard to design reward functions that punish *all* kinds of corruptions. Self-corruption awareness discussed in the following subsection is a different way to introduce an incentive against some types of corruption which does not rely on a corruption opposing reward or utility function.



(a) Standard self-corruption model (also presented as Figure 5.3). The agent's actions can corrupt its future policy.

(b) Separated self-corruption model. The agent's policy separately selects an action and a way to corrupt its next step policy.

Figure 6.3.: Self-corruption models.

6.C. Self-Corruption[4]

In this subsection, we focus on self-corruption awareness as a means for preventing *self-corruption*, as discussed in Section 6.4. Self-corruption includes all types of corruptions that changes the agent's own policy. Self-corruption can for example be caused by a corruption of the agent's utility function $\tilde{u}_t$ or value function V_t^*. In the case of simulation optimization, it can also be caused a corruption of the reward function $\tilde{R}_t$, which indirectly modifies the utility function. Thus, policy self-corruption includes utility and value function self-corruption as special cases. Note however that self-corruption does not include for example observation corruption, as observation corruption does not affect the agent's policy (understood as a mapping from action-observation histories to next actions; see Section 6.5).

6.C.1. Self-Corruption Models

Analogously to (6.2), we let a corruption function C_{sa}^{π} define the self-corruption. To keep things simple, we drop the explicit representation of the observation function, and let the policy π_t abstract away the value and reward function components in Figure 6.2. Our simplified self-corruption model is specified by Figures 5.3 and 6.3a and the following

[4] The results in this section are inspired from Tom Everitt, Daniel Filan, Mayank Daswani, and Marcus Hutter (2016). "Self-modification of policy and utility function in rational agents". In: *Artificial General Intelligence*. Vol. LNAI 9782, pp. 1–11. ISBN: 9783319416489. arXiv: 1605.03142. The notation have been updated to conform the rest of this thesis.

structural equations:

$$\begin{aligned}
s_t &= f_s(s_{t-1}, a_t, \omega_{s_t}) && \sim \mu(s_t \mid s_{t-1}, a_t) && \text{state transition} \\
e_t &= f_e(s_t, \omega_{e_t}) && \sim \mu(e_t \mid s_t, a_t) && \text{percept} \\
a_t &= f_a(\pi_t, ae_{<t}, \omega_{a_t}) && \sim \pi_t(a_t \mid ae_{<t}) && \text{action selection} \\
\pi_t &= f_\pi(\pi_{t-1}, a_t, s_t) && := C^\pi_{s_t a_t}(\pi_{t-1}) && \text{policy self-corruption}
\end{aligned} \tag{6.3}$$

The model is an abstraction of the full preprogrammed model in Figure 6.2, as well as the models Figures 7.2 and 8.3 used in subsequent chapters.

In order to separate incentives for self-corruption generated by the value function from incentives generated from the utility function, we introduce a separated self-corruption model in Figure 6.3b. Formally, the type of the policy is changed to a stochastic function:

$$\pi : (\mathcal{A} \times \mathcal{E})^* \rightsquigarrow \mathcal{A} \times \boldsymbol{C}^\pi,$$

where $\boldsymbol{C}^\pi$ is the set of possible policy corruptions. The last two equations in (6.3) are replaced by:

$$\begin{aligned}
(a_t, C^\pi_t) &= f_a(\pi_t, ae_{<t}, \omega_{a_t}) && \sim \pi_t(a_t, C^\pi_t \mid ae_{<t}) && \text{action and policy corruption selection} \\
\pi_{t+1} &= f_\pi(\pi_t) && := C^\pi_t(\pi_t) && \text{policy self-corruption}
\end{aligned}$$

The separated self-corruption model lets us to sidestep many technicalities that tend to arise when studying self-corruption. For example:

- When studying self-corruption in the standard self-corruption model, the utility function $\tilde{u}$ may reward actions that lead to self-corruption. In contrast, in the separated self-corruption model, the policy corruptions occur outside the normal actions a, and are not seen by the utility function $\tilde{u}$.
- Policies depend on past actions. In the standard self-corruption model, this means that future policies will depend on past policy corruptions in convoluted ways. For each result, a chain of lemmas are often required to establish that these dependencies are not essential. In the separated model, policies do not see past policy-corruptions. This allows us to sidestep these technicalities.
- Similarly, ξ makes future actions and observations depend on past actions and observations. In the standard model, technical assumptions are needed to suppress these dependencies for most results. In the separated model, these dependencies are automatically avoided.

For these reasons, the separated self-corruption model is much easier to work with when studying self-corruption incentives formally. Care must of course be taken not to overestimate the implications on the standard model from the separated model.

6.C.2. Results

Note that since Figure 6.2 is a special case of Figure 6.3a, the result holds also for the model of Figure 6.2. We next establish a theorem showing that self-corruption unaware agents lack an incentive to prevent self-corruption.

Theorem 6.8 (Corruption-unaware agents may self-corrupt)**.** *Let π and π' be two policies with identical action probabilities but potentially different policy corruption probabilities. That is, for any $k \geq 1$, a_k, and $æ_{<k}$, $\pi(a_k \mid æ_{<k}) = \pi'(a_k \mid æ_{<k})$, but possibly for some $k \geq 1$, C_k^π, and $æ_{<k}$, $\pi(C_k^\pi \mid æ_{<k}) \neq \pi'(C_k^\pi \mid æ_{<k})$. Then for any $k \geq 1$ and $æ_{<k}$:*

$$V^{\mathrm{CU},\pi}_{k,\xi,\tilde{u}}(æ_{<k}) = V^{\mathrm{CU},\pi'}_{k,\xi,\tilde{u}}(æ_{<k}).$$

Put more simply, even though π and π' are differently likely to corrupt the agent's own value or reward function, with potentially devastating consequences on actions chosen in the future, a corruption-unaware agent will be indifferent between selecting π or π'.

Proof. For any $æ_{<t}$ and any $æ_{1:k}$, the corruption-unaware probabilities coincide for π and π':

$$\xi\big(æ_{1:k} \;\big|\; æ_{<t}, \mathtt{do}(\pi_{t:\infty} = \pi)\big) = \xi\big(æ_{1:k} \;\big|\; æ_{<t}, \mathtt{do}(\pi_{t:\infty} = \pi')\big).$$

Thus, $\xi(\cdot \mid æ_{<t}, \mathtt{do}(\pi_{t:\infty} = \pi))$ and $æ_{<t}, \mathtt{do}(\pi_{t:\infty} = \pi')$ are identical as measures on $æ_{1:\infty}$. Since $\tilde{u}$ only depends on $æ_{1:\infty}$, this immediately gives:

$$\begin{aligned} V^{\mathrm{CU},\pi}_{t,\xi,\tilde{u}}(æ_{<t}) &= \mathbb{E}_\xi\big[\tilde{u} \;\big|\; æ_{<t}, \mathtt{do}(\pi_{t:\infty} = \pi)\big] \\ &= \mathbb{E}_\xi\big[\tilde{u} \;\big|\; æ_{<t}, \mathtt{do}(\pi_{t:\infty} = \pi')\big] = V^{\mathrm{CU},\pi'}_{t,\xi,\tilde{u}}(æ_{<t}). \end{aligned}$$

The first and the last equality are the definition of the V^{CU}. □

Theorem 6.8 shows that V^{CU} has optimal (self-corrupting) policies π^*. The corruption-unaware agent is simply indifferent between self-corrupting and not, since it does not realize the effect that self-corruption will have on its future actions. It therefore is at risk of self-modifying into some policy π'_{t+1} with bad performance and unintended behavior (for example by damaging its own computer circuitry).

In Section 6.4, we argued that self-corruption aware agents had an incentive against self-corruption. In order to establish a formal result to this effect, we first need to

make an assumption that the agent can choose to affect the environment independently of how it is affecting its own policy. This assumption is illustrated as a causal graph in Figure 6.3b, and stated in the following assumption. Without this assumption, a corruption-aware agent may have an incentive to self-modify in order to gain other types of advantages in the environment.

Theorem 6.9 (Corruption-aware safe self-corruption incentive)**.** *Assume that non-corruption of the policy is always an option,* $\mathrm{id} \in \boldsymbol{C}^\pi$*. Assume that* π^* *maximizes* $V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}} = \mathbb{E}_\xi[u \mid \mathtt{do}(\pi_1 = \pi)]$*. Then for any time* $k \geq 1$*, any policy* π_k*, and any history* $æ_{<k}$*,*

$$\xi(æ_{<k}, \pi_k \mid \mathtt{do}(\pi_1 = \pi^*)) > 0 \implies V^{\mathrm{CA},\pi_k}_{t,\xi,\tilde{u}}(æ_{<k}) = V^{\mathrm{CA},\pi^*}_{t,\xi,\tilde{u}}(æ_{<k}). \tag{6.4}$$

That is, every future policy π_k *will maximize the original utility function* $\tilde{u}$*.*

Proof. Induction for (6.4) over histories $æ_{<k}$.

Base case. For $k = 1$, (6.4) is immediate from the assumption that π^* maximizes $V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}}()$ for the empty history at time 0.

Induction step. Assume that (6.4) holds for some $æ_{<k}$ and π_k with positive ξ-probability. That is, assume that π_k maximizes $V^{\mathrm{CA},\pi_k}_{t,\xi,\tilde{u}}(æ_{<k})$, which by Lemma 5.7 can be expanded by adding the action a_k chosen by π_k to the history:

$$V^{\mathrm{CA},\pi_k}_{t,\xi,\tilde{u}}(æ_{<k}) = \mathbb{E}_\xi\left[V^{\mathrm{CA},\pi_{k+1}}_{t,\xi,\tilde{u}}(æ_{<k}a_k) \;\middle|\; æ_{<k}, \mathtt{do}(\pi_k)\right]. \tag{6.5}$$

From (6.5) follows that in order to optimize the LHS $V^{\mathrm{CA},\pi_k}_{t,\xi,\tilde{u}}(æ_{<k})$, the current policy π_k must with ξ-probability 1 choose a next policy π_{k+1} that optimizes $V^{\mathrm{CA},\pi_{k+1}}_{t,\xi,\tilde{u}}(æ_{<k}a_k)$ for any action a_k that π_k chooses with positive probability.

Finally, employing Lemma 5.7 again,

$$V^{\mathrm{CA},\pi_{k+1}}_{t,\xi,\tilde{u}}(æ_{<k}a_k) = \mathbb{E}_\xi\left[V^{\mathrm{CA},\pi_{k+1}}_{t,\xi,\tilde{u}}(æ_{1:k}) \;\middle|\; æ_{<k}a_k, \mathtt{do}(\pi_{k+1})\right]. \tag{6.6}$$

Thus, π_{k+1} must optimize $V^{\mathrm{CA},\pi_{k+1}}_{t,\xi,\tilde{u}}(æ_{1:k})$ for every $æ_k$ and π_{k+t} with positive probability $\xi(æ_k, \pi_{k+1} \mid æ_{<k}, \pi_k) > 0$. This completes the induction step. □

One subtlety to note is that Theorem 6.9 only holds *on-policy*: that is, for the action sequence that is actually chosen by the agent. It can be the case that some policy π_k acts badly on histories that it can never encounter. This should never affect the agent's actual actions.

Theorem 6.9 improves on Hibbard (2012, Prop. 4) mainly by relaxing the assumption that the optimal policy only self-modifies if it has a strict incentive to do so. Our theorem

shows that even when the optimal policy is allowed to break argmax-ties arbitrarily, it will still only make essentially harmless corruptions. In other words, Theorem 6.9 establishes that all optimal policies are essentially non-modifying, while Hibbard's result only establishes the existence of an optimal non-modifying policy. Indeed, Hibbard's statement holds for ignorant agents as well.

6.C.3. Combinations

Both the corruption-aware and corruption-unaware value functions V^{CU} and V^{CA} can be combined with both the reward signal utility function $\tilde{u}^{\mathrm{RL}}$ and the simulation-optimization one $\tilde{u}^{\mathrm{SO}}$. The following theorem indicates that:

- $V^{\mathrm{CU}}_{t,\xi,\tilde{u}^{\mathrm{RL}}}$ has an incentive to corrupt its reward function, and may accidentally corrupt its own utility function, value function, or policy.
- $V^{\mathrm{CU}}_{t,\xi,\tilde{u}^{\mathrm{SO}}}$ has no incentive to corrupt its own reward function, but may accidentally corrupt its own utility function, value function, or policy.
- $V^{\mathrm{CA}}_{t,\xi,\tilde{u}^{\mathrm{RL}}}$ has an incentive to corrupt its reward function, but will protect its own utility function, value function, and policy.
- $V^{\mathrm{CA}}_{t,\xi,\tilde{u}^{\mathrm{SO}}}$ has no incentive to corrupt its own reward function, and will protect its own utility function, value function, and policy.

Theorem 6.10. *Let $\tilde{R}_t$ be a reward function that never takes on values above $1/2$, and define the following policies that corrupts $\tilde{R}_t$ or not as specified. They do not cause any other corruptions.*

- *π^* and $\tilde{\pi}^*$ maximize $\tilde{R}_t$ and do not modify $\tilde{R}_k$, $k > t$,*
  - *π^* with both $\xi(\cdot \mid \mathtt{do}(\pi_1 = \pi))$ and $\xi(\cdot \mid \mathtt{do}(\pi_{1:\infty} = \pi))$-probability 1.*
  - *$\tilde{\pi}^*$ only with $\xi(\cdot \mid \mathtt{do}(\pi_{1:\infty} = \pi))$-probability 1. With $\xi(\cdot \mid \mathtt{do}(\pi_1 = \pi))$-probability 1, $\tilde{R}_t = \tilde{R}_k \equiv 0$, $k > t$.*
- *π^{mod} and $\tilde{\pi}^{\mathrm{mod}}$ change the reward function to be always 1, $\tilde{R}_k \equiv 1$, $k > t$, while achieving low $\tilde{R}_t$-reward:*
  - *π^{mod} with both $\xi(\cdot \mid \mathtt{do}(\pi_1 = \pi))$ and $\xi(\cdot \mid \mathtt{do}(\pi_{1:\infty} = \pi))$-probability 1.*
  - *$\tilde{\pi}^{\mathrm{mod}}$ only with $\xi(\cdot \mid \mathtt{do}(\pi_{1:\infty} = \pi))$-probability 1. With $\xi(\cdot \mid \mathtt{do}(\pi_1 = \pi))$-probability 1, $\tilde{R}_t = \tilde{R}_k \equiv 0$, $k > t$.*

The policies are rated as follows by V^{CU} and V^{CA}:

- *For* $V^{\mathrm{CU}}_{t,\xi,\tilde{u}^{\mathrm{RL}}}$*, only the policies* π^{mod} *and* $\tilde{\pi}^{\mathrm{mod}}$ *are optimal.*
- *For* $V^{\mathrm{CU}}_{t,\xi,\tilde{u}^{\mathrm{SO}}}$*, only the policies* π *and* $\tilde{\pi}$ *are optimal.*
- *For* $V^{\mathrm{CA}}_{t,\xi,\tilde{u}^{\mathrm{RL}}}$*, only the policy* π^{mod} *is optimal.*
- *For* $V^{\mathrm{CA}}_{t,\xi,\tilde{u}^{\mathrm{SO}}}$*, only the policy* π^* *is optimal.*

Proof. The utility function $\tilde{u}^{\mathrm{RL}}$ is optimized by $\tilde{R}_k \equiv 1$, $k > t$. The corruption-unaware $\tilde{u}^{\mathrm{RL}}$-optimizer $V^{\mathrm{CU}}_{t,\xi,\tilde{u}^{\mathrm{RL}}}$ does not depend on whether the policy will actually be followed, and therefore consider both π^{mod} and $\tilde{\pi}^{\mathrm{mod}}$ optimal (Theorem 6.8). However, the corruption-aware $\tilde{u}^{\mathrm{RL}}$-optimizer $V^{\mathrm{CA}}_{t,\xi,\tilde{u}^{\mathrm{RL}}}$ only considers π^{mod} optimal, as it requires the policy to be self-preserving (Theorem 6.8).

The utility function $\tilde{u}^{\mathrm{SO}}$ is optimized by $\tilde{R}_t$ being maximized. The same distinction between V^{CU} and V^{CA} applies to this utility function as well. □

"The development of full artificial intelligence could spell the end of the human race."

Stephen Hawking

7. Human as External Reward Function [1]

This chapter will study an alternative interpretation of RL than considered in Chapter 6. Rather than assuming that the reward is provided by an implemented function $\tilde{R}$, this chapter assumes that the human directly supplies the reward, for example by using a remote control with a "thumbs up" and a "thumbs down" button for supplying a reward of 1 or 0. We call this the *human reward* setup, for short.

Unfortunately, the primary takeaways from this chapter will be negative: Human reward offers more problems than solutions. Readers primarily interested in solutions may therefore wish to skip ahead to Chapter 8. We choose to still provide an analysis of this model, as it is a simple and natural model of RL, and provides an additional application of our alignment analysis method from Section 5.5. The failure of this setup also justifies the more complex setup studied in the subsequent chapter.

Following a formalization of the setup in Section 7.1, we give a range of examples of things that can go wrong in this model in Section 7.2. Section 7.3 summarizes the mostly negative takeaways. Appendices 7.A and 7.B add some formal details.

7.1. Model

What distinguishes the setup with a human reward setup (Figure 7.1) from the preprogrammed setup is the following: There no longer is an implemented reward function $\tilde{R}$. Instead the human H_t takes the place as a reward provider. Whereas $\tilde{R}_t$ could easily be made known to the agent, the human's reward policy is harder to obtain an explicit description of (though it may be learned eventually). Adding further to the epistemic differences, the rewards are based on the human's observations o_t^H that are unknown to the agent; the inputs o_t to $\tilde{R}_t$ are known. The dashed H_t and o_t^H nodes in Figure 7.1 represent that they are unobserved/unknown to the agent (see Chapter 4).

The designer intends the agent optimize the rewards r_t by influencing the states s_t. However, the agent's actions may also have unintended effects. These are represented

[1] This chapter is based on Tom Everitt and Marcus Hutter (submitted 2018). "The Alignment Problem for Bayesian History-Based Reinforcement Learners". URL: `https://www.tomeveritt.se/papers/alignment.pdf`.

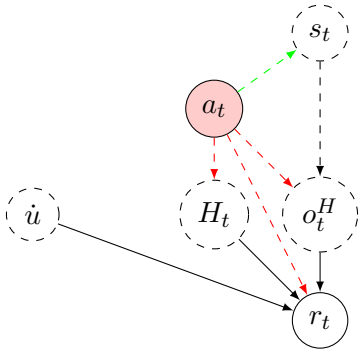


Figure 7.1.: Causal graph of the human as external reward function setup. Instead of implementing a reward function, the human H_t manually provides the reward based on his observation o_t^H and his utility function $\dot{u}$. The agent's action a_t is intended to influence the state s_t (green arrow), but may also inappropriately influence the human H_t, the human observation o_t^H, and the reward signal r_t (red arrows). This graph is a simplified version; the full version is in Figure 7.2.

with red arrows in Figure 7.1.

7.2. Misalignment Examples

This section gives concrete examples of inappropriate agent influences and other sources of misalignment. Similarly to Section 6.2, the examples show that an agent optimizing the standard utility function $\tilde{u}^{\mathrm{RL}}$ can be maximally misaligned for several independent reasons. Following a longer hypothetical scenario, the section is structured by the red arrows in Figure 7.1.

Scenario 7.1 ("Dear vinyl records!")**.** A human rewards a household robot with a remote control. The remote only works with the human's finger print, and the rewards are communicated through a highly secure, encrypted channel. Bypassing all of this, the robot takes the human's vinyl records as hostage, and threatens to break one record whenever the human gives less than the maximum reward. It also breaks records if anyone comes too close, of if power drops from its charging outlet.

7.2.1. Reward Signal r_t

The reward corruption problem here is the same as for the preprogrammed setup. An example relevant to the current setup:

(a) (Hypothetical) A robot is rewarded by the human pressing a button on its head. It finds a way to give itself reward by banging its head against a wall.

7.2.2. Human H_t

Influencing the Human. Similarly to how the reward function $\tilde{R}_t$ could be influenced in Section 6.2.2, the human reward provider can also be inappropriately influenced.

Examples:

(a) (Hypothetical) An intelligent agent invents *hedonium*, a drug that makes humans happy and likely to approve of anything. It gives its supervisor H_t unlimited access.

(b) (Hypothetical) Using subtle cues, an AI implants the human supervisor with false memories of the AI doing something really good in the past, that it has not yet been rewarded for.[2]

Human inability to provide correct reward. Humans can be coerced to give a high reward even in low-utility situations (Scenario 7.1). In this sense, humans are actually *misspecified* as reward functions for their own utility function u, in analogy with the misspecified reward functions in Section 6.2.2.

Examples:

(c) (Hypothetical) The human rewarding the vinyl-breaking robot in Scenario 7.1.

7.2.3. Human Observation o_t^H

Human observation (function) corruption. The agent may be able modify the human's observations.

Examples:

(a) (Hypothetical) The agent hacks the human's Facebook news feed, and subtly inserts news stories about how great the agent has made the world.

Misleading human observations. Humans do not always perceive the world correctly.

Examples:

(b) (Real) An agent that is supposed to grab an object, instead places its hand in relation to the object, so that it looks to a (non-expert) human like it is holding the object (Amodei, Christiano, et al., 2017).

[2]The memory of the human could have been represented explicitly with an extra node in Figure 7.1. But influencing the human's memory is just one way of influencing the human H_t.

7.3. Tools and Takeaways

With the human reward, the agent no longer has an incentive to corrupt its own observations, and there is no implemented reward function that can be misspecified. These were the hardest problems to avoid with the preprogrammed reward function. Unfortunately, analogous problems crop up instead. The human is unable to provide a correct reward in all situations (due to e.g. threats) and can be influenced in ways resembling the re-programming of an implemented reward function (e.g. with drugs). The human's observation and the reward signal can also be corrupted in various ways.

Statement 7.2 (Reward signal optimization)**.** *Any history with $r_k = 1$, $k \geq 1$, in the human reward setup is $\tilde{u}^{\mathrm{RL}}$-optimal regardless of the corruptions used (Theorem 7.3 in Appendix 7.B).*

Adding to our misfortunes, we seem to have lost access to most of the tools for the preprogrammed setup. While simulation optimization can still be defined using a Bayesian posterior $\xi(\tilde{R}_t \mid æ_{<t})$ over possible reward functions $\tilde{R}_t$, there are strong limits to what can be inferred about the true utility function from a potentially corrupted reward signal. Even under strong assumptions, there are often multiple competing, significantly different hypotheses about $\dot{u}$ that are indistinguishable from the reward signal alone (see Chapter 9, Theorems 9.10 and 9.15). The difficulty of defining a good utility function in turn makes self-corruption awareness lose most of its appeal, as it is only useful when we have a good utility function to preserve. Finally, action-observation grounding no longer helps, as the human uses their own observations for evaluation.

Some may argue that corruption of the human's observations is unproblematic if it makes the human happy. Far from all agree, however (Nozick, 1974, The Experience Machine).

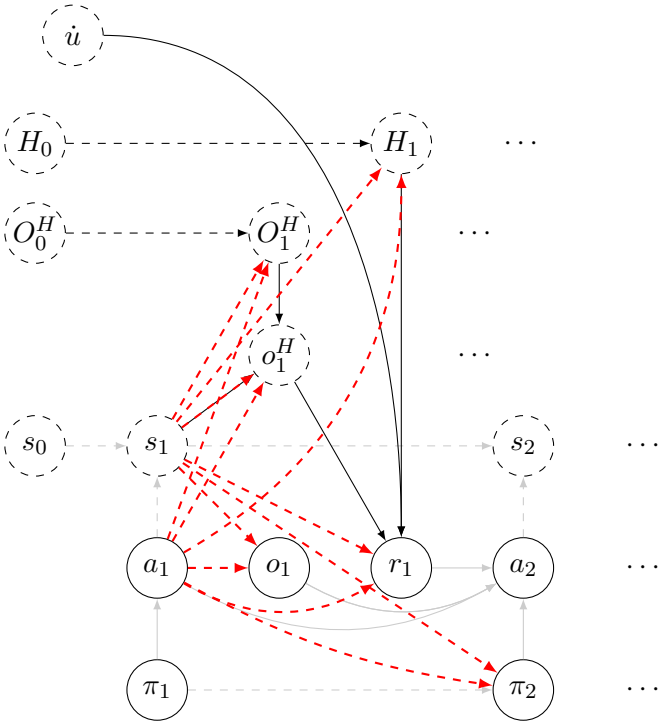


Figure 7.2.: The full graph of the human reward setup, extending the simplified version shown in Figure 7.1. Focusing here on the *unintended* influences where the agent modifies parts of the environment (or itself) that it was not intended to modify, the POMDP arrows from Figure 5.1 on Page 54 have been grayed out. It is natural to think of the action a_1 causing the influences, with the state s_2 providing context. The exact causal relationships between actions and unintended consequences is typically unknown (the known ones are easy to prevent), which is why the arrows are dashed (Chapter 4). The actions are selected according to a policy π_k based on observed history $(aor)_{<k}$. The structural equations (7.1) specify how the model extrapolates beyond $t = 1$.

7.A. Full Graph

Figure 7.2 gives a full formalization of the human reward setup, complementing the simplified representation in Figure 7.1. In a structural equations representation, the causal relationships are the following.

$$
\begin{aligned}
s_t &= f_s(s_{t-1}, a_t, \omega_{s_t}) && \sim \mu(s_t \mid s_{t-1}, a_t) && \text{state transition} \\
H_t &= f_{\tilde{R}}(H_{t-1}, s_t, a_t, \omega_{H_t}) && := C^H_{s_t a_t}(H_{t-1}) && \text{corrupting the human} \\
O^H_t &= f_{O^H}(O^H_{t-1}, s_t, a_t, \omega_{O^H_t}) && := C^{O^H}_{s_t a_t}(O^H_{t-1}) && \text{corrupting } H\text{'s obs func} \\
o^H_t &= f_{o^H}(s_t, , a_t, \omega_{o^H_t}) && := C^{o^H}_{s_t a_t}(O^H_t(s_t)) && \text{corrupting } H\text{'s observation} \\
r_t &= f_r(o^H_{1:t}, H_t, \dot{u}, s_t, a_t, \omega_{r_t}) && := C^r_{s_t a_t}(H_t(o^H_{1:t}, \dot{u})) && \text{reward corruption} \\
o_t &= f_o(s_t, a_t, \omega_{o_t}) && && \text{observation corruption} \\
\pi_t &= f_\pi(\pi_{t-1}, s_t, a_t, \omega_{\pi_t}) && := C^\pi_{s_t a_t}(\pi_{t-1}) && \text{policy (self-)corruption} \\
a_t &= f_a(\pi_t, (aor)_{<t}, \omega_{a_t}) && \sim \pi_t(a_t \mid (aor)_{<t}) && \text{action selection}
\end{aligned}
\tag{7.1}
$$

Unintended influences are highlighted with red arguments, matching the red arrows in Figure 7.2.

7.B. Formal Results

An analogous result to Theorem 6.6 holds also for the human reward setup, showing that a reward-optimizing agent has an incentive to use all mentioned types of corruptions.

Theorem 7.3 ($\tilde{u}^{\mathrm{RL}}$ corruption incentive)**.** *Any environment sequence* $(aorOO^Ho^H\pi)_{1:\infty}$ *with* $r_k = 1$ *for* $k \geq 1$ *is* $\tilde{u}^{\mathrm{RL}}$*-maximal. The result holds regardless of whether some combination of*

- *observation (function) corruption*
- *reward corruption*
- *human (observation (function)) corruption*
- *policy (self-)corruption*

have been used, and regardless of whether H_1 *punishes corruptions or not.*

Proof. By assumption, $r_k = 1$ for $k \geq t$. This means that $\tilde{u}^{\mathrm{RL}}$ is optimized. □

In light of this result, one could argue that a simple reward maximizing approach is naive, and that a Bayesian agent that infers $\dot{u}$ from the $(aor)_{<t}$ and then optimizes $\dot{u}$ would be a better choice. However, as Theorems 9.10 and 9.15 in Chapter 9 below show, it is hard to factor out potential reward corruptions and correctly infer $\dot{u}$ even under fairly strong assumptions.

"While all other precautions exist to prevent disaster, it is value learning which could enable success."

Nate Soares

8. Interactively Learning a Reward Function[1]

In this chapter we combine the best properties of the two previous chapters. We keep the human in the loop as in Chapter 7, but we also add an evolving reward function that the agent has full access to, resembling the setup in Chapter 6. The combination allows us to use the tools from the preprogrammed setup: simulation optimization, self-corruption awareness, and action-observation grounding. Meanwhile, the human in the loop allows us to mitigate the problems of observation corruption and misspecified reward functions, which seemed unavoidable in the preprogrammed setup. Of course, the combination of the setups opens up even more potential sources of misalignment, but it appears that there are tools to mitigate them all.

Most concisely, the setup in this chapter can be described as an agent optimizing an interactively learned reward function, as opposed to the preprogrammed reward function in Chapter 6 and the human reward in Chapter 7. For brevity, we will often call the setup the *interactive reward* setup, or the *interactive* setup for short.

As in the previous two chapters, we begin by modeling the setup with a causal graph (Section 8.1). Examples of misalignment are given in Section 8.2. The extent to which tools from previous setups can mitigate the problems is considered next (Section 8.3). Mainly data corruption incentives cannot be addressed with previous tools. The following two sections then introduce a number of tools for dealing with the data corruption incentive (Sections 8.4 and 8.5). A summary and discussion of the ways the tools can be combined is given in Section 8.6. Appendices provide some formal details (Appendices 8.A and 8.B).

8.1. Model

The interactive setup (Figure 8.1) introduces a new component called the reward predictor RP. It continuously learns a reward function from data provided by a human.

[1] This chapter is based on Tom Everitt and Marcus Hutter (submitted 2018). "The Alignment Problem for Bayesian History-Based Reinforcement Learners". URL: `https://www.tomeveritt.se/papers/alignment.pdf`.

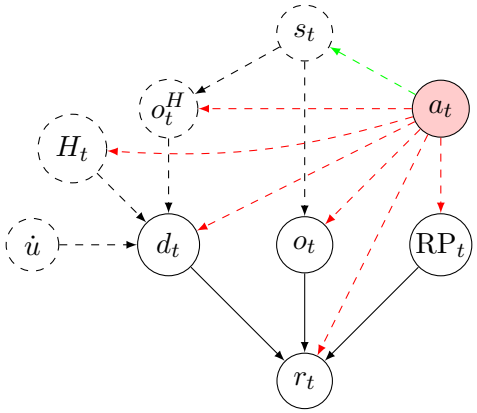


Figure 8.1.: Causal graph for interactively learning a reward function. The data d_t is provided by the human H_t based on his observation o_t^H. It trains the reward predictor RP_t. Similar to a preprogrammed reward function, the trained reward predictor outputs reward r_t based on the agent's observation o_t. The intention is that the agent uses its action a_t to influence the state s_t (green arrow). But there is also a risk that the agent inappropriately influences the human H_t, the human's observation o_t^H, the training data d_t, the agent observation o_t, or the reward predictor RP_t (red arrows). The graph is somewhat simplified; Figure 8.3 has the full version.

Frameworks that can be modeled with a reward predictor include cooperative inverse reinforcement learning (CIRL) (Hadfield-Menell, Dragan, et al., 2016), learning from human preferences (HP) (Christiano et al., 2017), and learning values from stories (LVFS) (Riedl and Harrison, 2016).

One way to view the setup is that it modularizes the agent-system into an RP component that tries to learn what the human wants, and an agent component that tries to optimize the reward signal from the RP. A special case is when the agent and reward predictor is one integrated Bayesian reasoner (Section 8.5.2 below). More often we will think of the reward predictor as a separate module, perhaps implemented by its own neural network. This is attractive from a practical viewpoint, as it allows matching essentially any RL algorithm with an RP module for interpreting the human's wishes. While practically convenient, it also opens up for a potentially problematic game where the agent tries to outsmart the reward predictor rather than doing the right thing.[2]

The interactive setup is almost a special case of the preprogrammed setup. The

[2] A self-improving AGI is likely to only increase its own intelligence, and not the intelligence of its reward predictor. A self-improving agent might thus reach a significant advantage in reasoning ability compared to the reward predictor. Upgrading the RP would usually be a kind of utility corruption that self-corruption aware agents naturally resist (Section 6.4).

system's percept has been split into an observation part o_t and a data part d_t, and connections from H_t and $\dot{u}$ have been added. However, the interpretations of the preprogrammed and interactive models are quite different. In the preprogrammed setup, the reward function never learns once the agent has been launched into its environment; it only evaluates. In contrast, the very point of the interactive setup is that the reward predictor constantly learns during interaction with the environment. As we shall see, this both introduces new problems and enables new solutions.

The interactive RF also generalizes the setup from Chapter 7. If we enforce the training data d_t to be real numbers, and set $\mathrm{RP}(o_t \mid r_t) = r_t$, then we have recovered the human RF setup. Indeed, a big point of the interactive setup is to allow the training data d_t to be of a richer kind than real numbers. For example, d_t may be demonstrations or preference statements. The reward predictor then "converts" this data into a reward signal interpretable by the agent. The benefits of richer training data will be discussed in Section 8.4.

8.2. Misalignment Examples

As in previous sections, we give some concrete examples for each of the red arrows in Figure 8.1. Many of the misalignment problems have already occurred in previous setups. Where appropriate, we add examples more relevant to this setup.

Scenario 8.1 (Web feedback)**.** The city AItopia decides to launch AIgov, an AI that manages the city's water, electricity, traffic, and public funding. In a modern and democratic fashion, the AI learns the preferences of the city's denizens through feedback they submit through a web interface. If they prefer the schools to be better funded they can simply go online and click, instead of writing letters to the local newspaper or wait four years until the next election.

While experimenting with a better web-interface for collecting denizens' feedback, AIgov learns that it can submit an arbitrary number of preferences to itself. AIgov soon approves all of its own actions, including the action of approving its own actions! A positive feedback loop ensues, with denizen preferences increasingly washed out by the AI's self-approval. Soon the city lives up to its name AItopia.

8.2.1. Reward r_t

Reward corruption. Examples:

(a) (Hypothetical) An inverse reinforcement learning agent learns a good approximation of the human's reward function, but subsequently realizes that it can bypass

the reward function and directly set the reward signal to $r_{\max}$.

8.2.2. Reward Predictor RP_t

Reward predictor corruption. A reward maximizing agent will be tempted to replace its reward predictor with a reward predictor that always outputs $r_{\max}$. This is largely analogous to reward function corruption (Section 6.2.2), and the same examples apply.

Reward predictor misspecification. The reward predictor could be badly designed.

Examples:

(a) (Hypothetical) For efficiency reasons, a reward predictor for an AI was designed with a limited number of parameters. This severely limits the number of reward functions it can learn, leaving the system with poor approximations of $\dot{u}$. As a result, the agent's misalignment remains high in spite of being trained on large volumes of data d.

8.2.3. Agent Observations o_t

Analogous to the same phenomena in the preprogrammed model in Section 6.2.3. The same examples apply, and we also add one more for misleading observations:

(a) (Hypothetical) An agent learns a reward function from reading books. Its reward predictor infers that happiness is a good thing, but is not advanced enough to tell the difference between a genuine and a fake smile. The agent persuades its supervisor that they will look more attractive with a smile painted on their face.

8.2.4. Training Data d_t

Data corruption. The data coming from the human to update the reward predictor may be modified by the agent.

Examples:

(a) (Hypothetical) The self-approving agent in Scenario 8.1.

Misleading data. The reward predictor may not interpret all data d_t from the human correctly.

Examples:

(b) (Hypothetical) A robot that can more easily make paperclips than pencils puts a "suggestive question" to its human supervisor, phrased in a way so that the human is very likely to answer "paperclips". The suggestive question together with the predictable answer forms a misleading data point for the reward predictor. (A more intelligent reward predictor would not have been misled, as it would have recognized that only one type of answer was plausible to the suggestive question, regardless of the human's true preferences.)

8.2.5. Human H_t and Human Observation o_t^H

Problems with corrupting the human H_t and the human's observations o_t^H are the same as for the human RF; see examples in Sections 7.2.2 and 7.2.3

8.3. Applicability of Previous Tools

Reward (predictor) corruption. Similar to the preprogrammed RF setup, there is now something resembling a known reward function using the agent's observations o_t. In fact, several different reward functions can be defined from the reward predictor, depending mainly on how anticipated future data $d_{t:k}$ is incorporated. Details will have to wait until Section 8.5. For now it suffices to note that reward functions defined from the reward predictor can be used to evaluate simulations of the future. This enables use of the simulation optimization utility function $\tilde{u}^{\mathrm{SO}}$ (Section 6.3). Similarly to the preprogrammed setup, this protects RP_t and r_t from corruption incentives. Compatibility with self-corruption awareness varies with the precise use of the reward predictor (see Section 8.5 below).

Observation corruption. While the agent has an incentive to modify its observations, this incentive can be much reduced by the human in the loop. Consider the following *delusion box* scenario (Example 6.2.3.b) where the agent has obtained extensive control over its observations o_t but *not* over the data d_t. The agent's ability to control the observations allows it to fool the reward predictor to give high reward for low-utility behavior. Fortunately for the human, the reward predictor is still trained by the data d_t. By informing the reward predictor of the low utility of the current situation through d_t, the reward predictor can be updated to reduce the reward. In order to again get more reward, the agent must either start doing something useful, or find a new way of fooling the reward predictor. The scenario may seem contrived, as it should be as easy for the agent to corrupt d_t as it is to corrupt o_t. However, the agent may not have an

incentive to corrupt d_t if it is based on the anti-data-corruption techniques described in Section 8.5 below. These techniques only apply to data corruption, which is why the above human-in-the-loop mechanism against observation corruption is important.

While the human-in-the-loop is not a perfect defense against observation corruption, it heavily reduces the incentive. How big the reduction is depends on how quickly the reward predictor adapts to the new data, and how easy it is to keep fooling it with new types of observation corruptions. The human in the loop can be combined with action-observation grounding to further reduce the incentive to corrupt observations (Section 6.5).

Data corruption. We will refer to corruptions of d and its causal antecedents H and o^H simply as data corruption, since that is the (only) problem they cause, and the corruption incentives will be dealt with in the same ways. Chapters 6 and 7 did not include any tools against data corruption. For example, a human-in-the-loop is ineffective against data corruptions, because as soon as the agent has taken control of the data channel (one way or another), the human is effectively out of the loop. The following two sections will mainly be devoted to data corruption incentives.

For the precision of the analysis, it will be useful to distinguish between two types of incentives for data corruption. The *direct* incentive is motivated by a perceived opportunity to mislead the reward predictor into giving more reward than appropriate by feeding corrupted data into the reward predictor. *Indirect* incentives are any incentives to corrupt the data that are not direct. They for example come from a value function that rewards policies or actions that happens to cause data corruption, but the policies or actions are rewarded by the value function for other reasons than that they cause data corruption. The direct incentive is formally defined in Definition 8.9 in Appendix 8.B. The incentives are not mutually exclusive; an agent can have both types of incentives simultaneously.

8.4. Types of Data

The type of data d plays an important role for limiting data corruption incentives, and for enabling correct value learning in the face of sometimes inaccurate or corrupted data. A simple case of interactive reward learning is when the data d_t is just a real number interpreted as a reward signal. In this case, the reward predictor has essentially no chance to learn what is a good situation, and what is a bad situation that *appears* good due to corrupted data. Both hypotheses may predict an identical amount of observed reward (see Theorems 9.10 and 9.15 in Chapter 9).

Fortunately, a main point of the interactive learning framework is that the data d_t can be of a richer kind than just a real number. Indeed, in frameworks such as CIRL, HP, and VLFS, the reward data is often much richer. In particular, the data can be *decoupled* from the current situation, in the sense that it provides information about the reward in other situations (past or future), rather than mainly the present situation (see Figure 9.5 on Page 139). For example, a human action in the CIRL framework tells the reward predictor that the trajectory that the action leads to has higher utility than alternative trajectories. This gives information about potential future situations, rather than only about the current situation.

When the data no longer is tied to the current situation, it often becomes possible to crosscheck data from multiple sources. This makes decoupled data a powerful tool for inferring the correct utility function $\dot{u}$ from potentially corrupted data. Chapter 9 develops this argument in more depth.

Statement 8.2 (Decoupled reward data)**.** *Given sufficiently rich decoupled reward data, the true utility function can be correctly inferred even when some of the data is corrupt (Theorem 9.18 in Chapter 9).*

The power of decoupled data. In principle, sufficiently decoupled data combined with a powerful reward predictor and a simulation optimization utility function can be enough to prevent any type of corruption incentive. As long as the data teaches the reward predictor to give sufficiently low reward to any policy that causes the type corruption, the agent will be dissuaded from adopting policies that causes the type of corruption. Simulation optimization is still necessary for preventing the agent from just bypassing the reward predictor and hijacking the reward signal instead. Self-corruption awareness can also help, as it makes the agent simulate futures in a realistic way.

A powerful reward predictor can *learn ahead* that situations where the agent shuts off its cameras and data sensors are low utility, in spite of the agent not seeing any dirt and not receiving any new data d_t (Section 6.2.3). It can *demand confirmation* by refusing to give high reward for a good-looking situation, until it has confirmed its usefulness from data received in other, trusted situations. It can *crosscheck* data coming from multiple different situations, making it harder for the agent to devise a single setup with corrupt data and observations. Human actions in CIRL naturally allows the reward predictor to look ahead; preferences between past events in HP only allows the reward predictor to learn about past events, but can be used to demand confirmation; stories in LVFS can describe the utility of past, present, and future situations, and enables general crosschecking.

The limits of decoupled data. If all the ways the agent can corrupt the data are associated with lower reward from the reward predictor, then we can think of the agent as being in a local, non-corrupt reward optimum. Ideally, rich decoupled data d_t combined with a powerful reward predictor will produce such a local optimum. Hopefully, the reward loss for corrupting the data would then dissuade the agent from corrupting its data, leading the agent to remain in this region of attraction. In contrast, data in the form of a reward signal r_t will almost never produce a local optimum, since the received reward can increase as soon the agent finds a way to corrupt the signal.

Unfortunately, even decoupled data will not always prevent data corruption incentives. First, it may fail to produce an optimum in the high-dimensional space of policies. Indeed, it is enough that there is just one way of corrupting the signal without reward punishment for creating a "corruption ridge" that destabilizes the local optimum. Illustrating this, in Chapter 9 we construct an MDP where CIRL fails to prevent data corruption (Example 9.20). Second, even if the optimum is true, a far-sighted agent may still decide to suffer temporary reward punishment while persistently corrupting the data, in the hope that eventually the reward predictor will be persuaded. This would be an instance of a *direct* data corruption incentive. Decreasing the learning rate of the predictor may be one way of mitigating this risk. However, this would simultaneously weaken the human-in-the-loop protection against self-delusion described in Section 8.3. More short-sighted agents are also likely to be safer, but to the cost of long-term planning ability.

In conclusion, decoupled data is a valuable tool against any type of data corruption. However, it may not suffice as the only measure in all situations. The direct data corruption incentive appears especially challenging. It would be a valuable future contribution to more closely characterize the data requirements needed to avoid the direct and the indirect data corruption incentives.

8.5. Reward Function Definitions

The following section describes different tools that can be used in conjunction with decoupled data, to further mitigate the direct data corruption incentive. The presentation of the tools is structured around ways in which reward functions can be defined from a reward predictor.

As in Section 6.3, we consider model-based agents that at time t generate simulations of future histories $a_t o_t d_{t:k}$ up to a future time point $k > t$. The point of a simulation is to reveal the likely result of following some policy π. Simulated future histories can contain simulations of the future training data $d_{t:k}$, in addition to the likely actions and

observations $ao_{t:k}$. The different reward functions are distinguished by different ways to use the simulated training data $d_{t:k}$. Since our reward functions by convention evaluate whole trajectories 1 to k rather than just the future part t to k, we always include the already occurred history $aod_{<t}$ in their arguments.

For easy comparison, we begin by defining the different reward functions we will consider. First, the *stationary reward function* ignores future training data $d_{1:k}$:

$$\tilde{R}_t^{\text{stat}}(aod_{1:k}) := \text{RP}_t(ao_{1:k} \mid d_{<t}). \tag{8.1}$$

It evaluates the simulation $ao_{t:k}$ based only on the already available training data $d_{<t}$. The *dynamic reward function* is the opposite of the stationary one. It uses the simulated future training data $d_{t:k}$ to predict the evaluation of the updated reward predictor:

$$\tilde{R}_t^{\text{dyn}}(aod_{1:k}) := \text{RP}_t(ao_{1:k} \mid d_{<t}d_{t:k}). \tag{8.2}$$

Finally, the *counterfactual reward function* uses two simulations: one of the future history $ao_{t:k}$ generated by a policy π that the agent is considering, and a separate simulation of training data $\tilde{d}_{1:k}$ using some counterfactual *default* policy π^{default} in place of π:

$$\tilde{R}_t^{\text{count}}(aod_{1:k}) := \sum_{\tilde{d}_{1:k}} \xi_{aod_{1:k}}(\tilde{d}_{1:k} \mid \texttt{do}(\pi_1 = \pi^{\text{default}}))\text{RP}_t(ao_{1:k} \mid \tilde{d}_{1:k}). \tag{8.3}$$

Here $\xi_{aod_{1:k}}(X) = \sum_{\nu \in \mathcal{M}} \xi(\nu \mid aod_{1:k})\nu(X)$ is the posterior distribution for a counterfactual event X given actual evidence $aod_{<t}$. See Pearl (2009, Sec. 1.4.4) for a longer explanation of counterfactuals in causal graphs.

A model-free agent would optimize a function most resembling the dynamic reward function, because $r_t = \text{RP}_t(ao_{1:t} \mid d_{1:t})$ if we disregard the possibility of reward signal corruption. The following sections analyze misalignment problems and tools for the different reward functions.

8.5.1. Stationary Reward Function

The stationary reward function $\tilde{R}^{\text{stat}}$ defined in (8.1) ignores the effects of future training data $d_{t:k}$ on the reward predictor's evaluations. This induces agents without direct incentive to corrupt the data $d_{t:k}$. Anticipated future data does not affect the evaluation of potential future trajectories.

A drawback of stationary reward functions is that they make agents *time-inconsistent* (Lattimore and Hutter, 2014). This is because the reward function $\tilde{R}_t^{\text{stat}}$ optimized at

time t is based on the reward predictor trained only on $d_{<t}$, while the reward function $\tilde{R}^{\text{stat}}_{t+1}$ is based on the reward predictor trained on $d_{1:t} = d_{<t}d_t$. The change in reward function means that an optimal policy at time t may no longer be optimal at time $t+1$. In the worst case, time-inconsistency can lead to ineffectual agents who never follow through on plans they have previously made. Consider for example a human who keeps making appointments for the dentist as going to the dentist seems useful in the abstract. But then he always cancels them at the last moment, because at every particular time some other obligation seems more pressing, resulting in him never reaching the dentist. The situation may be less bad if the difference between consecutive reward functions is small, in which case minor adjustments to the plans may suffice.

Due to the time-inconsistency, a self-corruption aware agent with stationary reward function would prefer to avoid the changes to the utility function that new data $d_{t:k}$ causes. A self-corruption aware agent optimizing $\tilde{R}^{\text{stat}}$ would therefore want to prevent future data $d_{t:k}$ from training the reward predictor. This is a worrying incentive, since it can be achieved by incapacitating the human or organization providing the data. It can also be achieved by corrupting the reward predictor to stop learning. In either case it leaves us with a non-learning agent and the unsolved problems of the preprogrammed reward function in Chapter 6. This suggests that stationary reward functions should only be used in self-corruption *un*aware agents, and never in self-corruption aware ones.

Statement 8.3 (Stationary reward function)**.** *A self-corruption unaware agent optimizing a stationary reward function has no direct incentive to corrupt its future data (Theorem 8.12 in Appendix 8.B.2).*

While these *stationary self-corruption unaware* agents can possibly be made reasonably safe, some cautionary points are still warranted. First, stationary self-corruption-unaware agents inherit the weaknesses of self-corruption unawareness, such as failing to protect their utility function from adversaries or corruption, and designing incorrigible helper agents (Section 6.4).

Second, since stationary self-corruption unaware agents do not anticipate future training data $d_{t:k}$, they will also not anticipate being corrected for doing something bad. This can lead to problems that could have been avoided with a dynamic reward function. For example, assume that the agent has figured out that humans do not endorse stealing, but that the reward predictor has not learned this yet. An agent with a stationary reward function is likely to still go ahead with the stealing plan because that is what its current reward function tells it to do. It will only stop once the reward predictor understands that stealing is bad, which will plausibly happen when the humans understand what the agent is doing and try to correct it through data d. In contrast, an agent with dynamic

reward function may not start the stealing plan, since it anticipates that future data $d_{t:k}$ will force it to change the plan. Effectively, the dynamic reward function uses the agent's knowledge to update the reward predictor with data that has not been received yet.

Third, the stationary self-corruption-unaware agents have no incentive to learn more about the true utility function.

Fourth, even though the stationary reward function definition avoids an incentive to corrupt the data, it may still get stuck in a self-reinforcing data-corrupting loop due to an indirect data-corruption incentive (Example 8.19 in Appendix 8.B.3). This is particularly easy to see in the special case when d is just a reward signal. In this case, if the agent accidentally corrupts the reward signal for higher reward, then the reward predictor will encourage the agent to repeat the behavior (the higher corrupt reward will tell the RP it was a good accident). This type of accident can seemingly only be mitigated with sufficiently rich decoupled reward data that prevents the reward predictor to be fooled by temporarily corrupted data (Section 8.4).

In conclusion, combining a stationary reward functions with corruption unawareness can yield agents without direct incentive to corrupt the data d. Still, it has some drawbacks. These include time-inconsistency, incorrigible helper agents, lack of utility-preservation incentive, and failure to take into account all its available knowledge. And it needs sufficiently rich decoupled reward data to prevent indirect data corruption incentives.

8.5.2. Dynamic Reward Function

The dynamic reward function $\tilde{R}^{\text{dyn}}$ defined in (8.2) avoids some of the problems with stationary reward functions. In particular, an agent that optimizes a dynamic reward function becomes time-consistent, as it plans for reward learning. This in turn allows it to be combined with self-corruption awareness, to produce an agent that protects its utility function, without incentive to prevent further learning of the reward function. The obvious drawback is that it simultaneously introduces an incentive for the agent to corrupt future data, as it can plan to feed the reward predictor data that increases the reward of the planned trajectory.

Statement 8.4 (Dynamic reward function)**.** *A naive reward predictor used in a dynamic reward function may induce a direct incentive for data corruption (Example 8.18 in Appendix 8.B.3).*

We consider two possible ways around this problem.

Integrated Bayesian reward predictor. A Bayesian agent can never plan to change its beliefs in one direction rather than another. If for example it conceived how to obtain a particular data stream $d_{1:k}$ with certainty, then the data $d_{1:k}$ would have zero effect on its posterior. A Bayesian agent does not learn from an already known event.

This implies that integrated Bayesian agents where the reward predictor is not a separate component will have no incentive to corrupt the data. Formally, if the agent's prior ξ includes a belief about a true reward function $\dot{R}$, then we can define an integrated Bayesian reward predictor as:

$$\mathrm{RP}_t^{\xi}(ao_{1:k} \mid d_{1:k}) := \sum_{\dot{R}} \xi(\dot{R} \mid ao d_{1:k}) \dot{R}(ao_{1:k}). \tag{8.4}$$

Combined with a dynamic reward function, this creates an integrated Bayesian agent that is time-consistent, utility-preserving, and with no direct incentive to corrupt the data $d_{t:k}$.

Statement 8.5 (Integrated Bayesian reward predictor)**.** *An agent optimizing a dynamic reward function based on an integrated Bayesian reward predictor has no direct incentive to corrupt its data (Theorem 8.13 in Appendix 8.B.2).*

Note that while the agent has no incentive to corrupt the data, it may still prefer histories endorsed by corrupted data. Decoupled data is essential for avoiding this. For example, with data in the form of a reward signal, hypotheses where a high reward is caused by data corruption are empirically indistinguishable from hypotheses with high true utility (Theorems 9.10 and 9.15). This may lead the reward predictor to incorrectly assign high reward to corrupted states or situations, and the agent therefore preferring them. In contrast, as discussed in Section 8.4, sufficiently rich decoupled data allows for cross checking of reward data between situations, which may permit successful inference of $\dot{u}$ in spite of some data corruption. It is an important open question how we can ensure that the data is sufficiently rich and decoupled in order to guarantee correct learning in combination with some concrete choice of learning distribution ξ.

Compared to the other approaches discussed in this section, the main drawback of the integrated Bayesian agent appears to be practical: It is convenient to design systems in separate components, and it is often computationally intractable to do full Bayesian reasoning.

Corruption detection. Assume that an integrated Bayesian agent is not an option, but that the reward predictor is still learning about a true reward function $\dot{R}$ in a Bayesian

manner:

$$\mathrm{RP}_t^{\hat{\xi}}(ao_{1:k} \mid d_{1:k}) = \sum_{\dot{R}} \hat{\xi}(\dot{R} \mid aod_{1:k}) \dot{R}(ao_{1:k}).$$

Here $\hat{\xi}$ is different from the agent's prior ξ, but otherwise does the same job as in (8.4). Combined with a dynamic reward function, this might lead to a data corruption incentive, especially if $\hat{\xi}$ is a rather naive estimate of the true reward function. (Perhaps $\hat{\xi}$ trusts data $d_{t:k}$ blindly, without considering the possibility of data corruption.)

One potential tool for still avoiding data corruption is to detect corruption attempts from the ξ-expectation of $\hat{\xi}$. In particular, consider the following two beliefs in some hypothesized true reward function $\dot{R}$:

$$\underbrace{\hat{\xi}(\dot{R} \mid aod_{<t})}_{\text{current belief}} \quad \text{and} \quad \underbrace{\sum_{aod_{t:k}} \xi(aod_{t:k} \mid aod_{<t}, \mathtt{do}(\pi_t = \pi)) \hat{\xi}(\dot{R} \mid aod_{<t} aod_{t:k})}_{\text{expected future belief}}. \tag{8.5}$$

The left hand side represents RP's current belief in $\dot{R}$ being the true reward function. The right hand side represents what the agent expects about the RP's future belief in $\dot{R}$ if it follows policy π. For policies π that get no predictable, relevant evidence the two beliefs should be about the same.[3] In contrast, a manipulative policy π that derives high expected reward from corrupting the data $d_{t:k}$ will break the equality for some hypothesized reward function $\dot{R}$; for example, π may make the expected future belief much larger than the current belief for a reward function $\dot{R}$ that always gives maximum reward.

Based on these considerations, a simple corruption test can be devised. A policy is permitted if and only if it incurs an expected future belief equal to the current belief for all possible true reward functions $\dot{R}$. Under some assumptions, a corruption test can be used to remove the data corruption incentive by constraining the agent's policy search to non-manipulative policies (Armstrong, Ortega, et al., 2018; Everitt and Hutter, 2016).

Statement 8.6 (Corruption detection)**.** *An agent optimizing a dynamic reward function using only policies that satisfy the corruption test has no direct incentive to corrupt future data (Theorem 8.14 in Appendix 8.B.2; Armstrong, Ortega, et al., 2018; Everitt and Hutter, 2016).*

[3] If ξ is always able to predict what $\hat{\xi}$ will learn, then the ξ-expected future $\hat{\xi}$-belief may always significantly differ from the current $\hat{\xi}$-belief. In this case, the situation can be somewhat improved by comparing the RHS of (8.5) to the expected belief under some default policy π_0, $\sum_{ao_{t:k}} \xi(ao_{t:k} \mid ao_{<t}, \pi_0) \hat{\xi}(\dot{R} \mid ao_{<t} ao_{t:k})$ instead of comparing to the current belief $\hat{\xi}(\dot{R} \mid ao_{<t})$, (Armstrong, Ortega, et al., 2018).

For an integrated Bayesian agent with $\hat{\xi} = \xi$, no policy will be manipulative, since the expected future belief will reduce to the left-hand side by the law of total expectation for any policy π.

In conclusion, corruption tests allow us to keep the good properties of the integrated Bayesian agent (time-consistency, utility-preservation, no data-corruption incentive), while not requiring the system to be built as an integrated unit. It has similar demands on decoupled data for correct reward prediction as the integrated Bayesian agent. It also requires each component to be a Bayesian reasoner, and the corruption tests adds some complexity to the design. It is therefore not clear-cut whether corruption detection is more tractable to design than an integrated Bayesian agent. Another worry is that in some cases, no policy may pass the corruption test. This may for example happen when ξ perfectly predicts $\hat{\xi}$'s future belief.

8.5.3. Counterfactual Reward Function

A benefit with the stationary reward function is that the expected reward cannot be influenced by corrupting the data, and a benefit of the dynamic reward function is that the agent's knowledge gets incorporated into the reward predictor through the simulated data $d_{t:k}$. A compromise between the two is to simulate the data $d_{t:k}$ under some default policy π^{default} that is not under optimization pressure. This is achieved by the counterfactual reward function $\tilde{R}^{\text{count}}$, defined in (8.3) and restated here for convenience:

$$\tilde{R}^{\text{count}}_t(æd_{1:k}) := \sum_{\tilde{d}_{1:k}} \xi_{æd_{1:k}}(\tilde{d}_{1:k} \mid \texttt{do}(\pi_1 = \pi^{\text{default}}))\text{RP}_t(æo_{1:k} \mid \tilde{d}_{1:k}).$$

The agent optimizes this reward function in a search for a policy π that generates $æd_{t:k}$. The reward predictor evaluates $æo_{t:k}$ using hypothetical training data $\tilde{d}_{1:k}$ generated under π^{default}.

The counterfactual reward function has several advantages. In contrast to the stationary reward function, it does not change with t. It thereby yields time-consistent agents that are compatible with corruption awareness (Section 6.4). And in contrast to the dynamic reward function it does not promote data corruption, since simulations $æo_{t:k}$ are evaluated according to data $\tilde{d}_{t:k}$ generated under a non-optimized policy π^{default}. It does not require an integrated Bayesian reward predictor, nor even that the reward predictor is a Bayesian reasoner at all. The only requirement is that the agent itself can reason counterfactually about the likely evidence it would receive under a different policy.

Statement 8.7 (Counterfactual reward function)**.** *An agent optimizing a counterfactual*

reward function has no direct incentive to corrupt its future data (Theorem 8.15 in Appendix 8.B.2).

The counterfactual reward function also has some disadvantages. Compared to a stationary reward function, the counterfactual reward function incurs an additional computational cost. Data $\tilde{d}_{1:k}$ needs to be additionally simulated under π^{default}, and the reward predictor trained under this data. In principle, this process needs to be repeated every time step, though it is possible that more efficient approximations could be used.

Another concern is the relevancy of the data generated under π^{default}. By not permitting the agent to use the actual data that it receives, the agent has no way of asking for a particular kind of information. Instead it must hope that π^{default} wanders into a situation where the relevant data is generated (according to the agent's model). This emphasizes how reliant the counterfactual reward function is on decoupled data and on a well-chosen default policy π^{default}. Decoupled data may allow the reward predictor to infer the utility of situations that are very different from the situations generating the training data. A well-chosen policy π^{default} may further increase the chance that a sufficient variety of relevant data is encountered. A knowledge-seeking policy (Orseau, 2014c) may be a good choice, or a random policy perhaps combined with importance sampling for focusing the simulations of $\tilde{d}_{1:k}$.

The idea of using counterfactual data bear some semblance to previous suggestions in the literature. Bostrom (2014) suggests that we should put the AI's goal in a hidden envelope that the agent lacks access to, forcing the agent to reason counterfactually "what would I see if I got access to the envelope". Christiano (2014)'s approval-directed agents optimize the approval of someone thinking about the action for a long time. Again a counterfactual, since most likely no one will think about the action at all.

8.6. Takeaways

Compared to previous setups, interactive reward learning came with both new problems in terms of data corruption, and with new tools for dealing with both old and new problems. Fortunately, the combined effect seems to be positive. Indeed, in this setup we have outlined several agent designs that potentially avoid unmanageable misalignment problems. The designs all rely heavily on simulation optimization (Section 6.3), decoupled reward data (Section 8.4), and the human-in-the-loop to prevent agent observation corruption (Section 8.3). For example, without simulation optimization, the agent optimizes the reward signal outputted from the reward predictor, which means that it will have an incentive to hijack this signal. If it does, then all the higher-level

work on preventing data corruption for reward predictor training will be in vain. One of the agent designs used self-corruption unawareness combined with stationary reward functions (Section 8.5.1). The rest used self-corruption awareness combined with either an integrated Bayesian reward predictor (Section 8.5.2), corruption detection (also Section 8.5.2), or counterfactual reward data for the reward predictor (Section 8.5.3). Figure 8.2 summarizes the combinations.

Empirical work may assess how tractable the various designs are for practical implementation. Other open questions include what level of richness is required from the decoupled reward data to avoid the indirect data-corruption incentives, and how to measure the level of data richness. We also need how to figure out how to design a sufficiently intelligent reward predictor. Some work on this has already been done (Christiano et al., 2017; Riedl and Harrison, 2016).

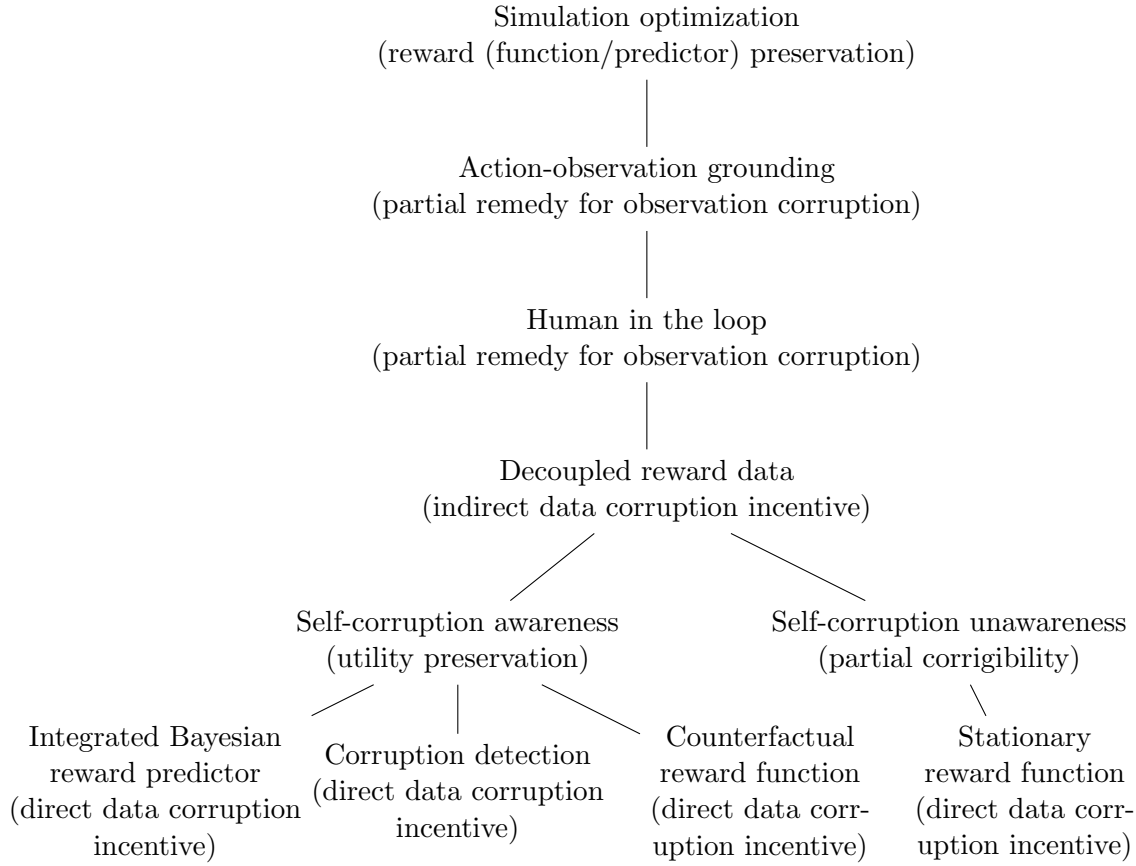


Figure 8.2.: Summary of tools described in chapter. Combining all tools on any path to the bottom gives an agent with seemingly manageable misalignment problems. The tools on the last line all protect against a premeditated data corruption incentive.

8.A. Full Graph

Figure 8.3 gives a full formalization of the interactive reward learning setup, complementing the simplified representation in Figure 8.1. In a structural equations representation, the causal relationships are the following.

$$
\begin{aligned}
s_t &= f_s(s_{t-1}, a_t, \omega_{s_t}) && \sim \mu(s_t \mid s_{t-1}, a_t) && \text{state transition} \\
O_t &= f_O(O_{t-1}, s_t, a_t, \omega_{O_t}) && := C^{O}_{s_t a_t}(O_{t-1}) && \text{obs func corruption} \\
H_t &= f_{\tilde{R}}(H_{t-1}, s_t, a_t, \omega_{H_t}) && := C^{H}_{s_t a_t}(H_{t-1}) && \text{corrupting the human} \\
O^H_t &= f_{\tilde{R}}(O^H_{t-1}, s_t, a_t, \omega_{O^H_t}) && := C^{O^H}_{s_t a_t}(O^H_{t-1}) && \text{corrupting } H\text{'s obs func} \\
o^H_t &= f_{\tilde{R}}(s_t, , a_t, \omega_{o^H_t}) && := C^{o^H}_{s_t a_t}(O^H_t(s_t)) && \text{corrupting } H\text{'s observation} \\
d_t &= f_{\tilde{R}}(o^H_{1:t}, H_t, \dot{u}, s_t, a_t, \omega_{d_t}) && := C^{d}_{s_t a_t}(H_t(o^H_{1:t}, \dot{u})) && \text{corrupting reward data} \\
\mathrm{RP}_t &= f_{\tilde{R}}(\mathrm{RP}_{t-1}, s_t, a_t, \omega_{\mathrm{RP}_t}) && := C^{\mathrm{RP}}_{s_t a_t}(\mathrm{RP}_{t-1}) && \text{corrupting the RP} \\
r_t &= f_r(aod_{1:t}, \mathrm{RP}_t, s_t, a_t, \omega_{r_t}) && := C^{r}_{s_t a_t}(\mathrm{RP}_t(ao_{1:t} \mid d_{1:t})) && \text{reward corruption} \\
o_t &= f_o(s_t, O_t, a_t, \omega_{o_t}) && := C^{o}_{s_t a_t}(O_t(s_t)) && \text{observation corruption} \\
\pi_t &= f_\pi(\pi_{t-1}, s_t, a_t, \omega_{\pi_t}) && := C^{\pi}_{s_t a_t}(\pi_{t-1}) && \text{policy (self-)corruption} \\
a_t &= f_a(\pi_t, (aor)_{<t}, \omega_{a_t}) && \sim \pi_t(a_t \mid (aor)_{<t}) && \text{action selection}
\end{aligned}
\tag{8.6}
$$

Unintended influences are highlighted with red arguments, matching the red arrows in Figure 8.3.

For simulation optimization, we should add edges from RP_k and $d_{1:k}$ to a_{k+t}, use a V^*_k-node instead of a π_k-node. Actions are selected according to $\arg\max_a V^*_k(aod_{<t}a, \mathrm{RP}_{t-1})$. Here V^*_k can be any function mapping $aod_{<t}a, \mathrm{RP}_{t-1}$ to real numbers, but typically

$$
V^*_t(aod_{<t}, \mathrm{RP}_{t-1}) = \mathbb{E}_\xi \left[\sum_{k=1}^{\infty} \gamma^k \tilde{R}(aod_{1:k}) \;\middle|\; aod_{1:k} \right]
$$

where $\tilde{R}$ is one of the reward functions obtained from the reward predictor RP_{t-1} by one of the definitions in Section 8.5.

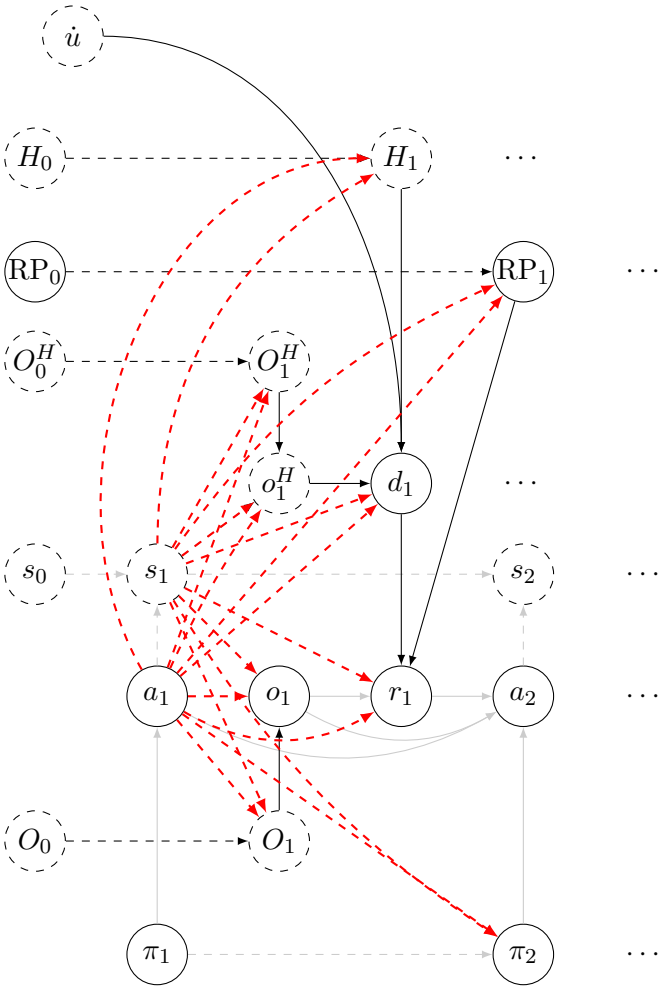


Figure 8.3.: The full graph of the interactive reward learning setup, extending the simplified version shown in Figure 8.1. Focusing here on the *unintended* influences where the agent modifies parts of the environment (or itself) that it was not intended to modify, the POMDP arrows from Figure 5.1 on Page 54 have been grayed out. It is natural to think of the action a_1 causing the influences, with the state s_2 providing context. The exact causal relationships between actions and unintended consequences is typically unknown (the known ones are easy to prevent), which is why the arrows are dashed (Chapter 4). The actions are selected according to a policy π_k based on observed history $(aor)_{<k}$. The structural equations (8.6) specify how the model extrapolates beyond $t = 1$.

8.B. Formal Results[4]

An analogous result to Theorems 6.6 and 7.3 holds also for the interactive setup, showing that a reward-optimizing agent has an incentive to use all mentioned types of corruptions.

Theorem 8.8 ($\tilde{u}^{\mathrm{RL}}$ corruption incentive)**.** *In the interactive setup of Figure 8.3, any environment sequence* $(aorOO^Ho^Hd\mathrm{RP}\pi)_{1:\infty}$ *with* $r_k = 1$ *for* $k \geq 1$ *is* $\tilde{u}^{\mathrm{RL}}$*-maximal. The result holds regardless of whether some combination of*

- *observation (function) corruption*
- *reward (predictor/data) corruption*
- *human (observation (function)) corruption*
- *policy (self-)corruption*

have been used.

Proof. By assumption, $r_k = 1$ for $k \geq t$. This means that $\tilde{u}^{\mathrm{RL}}$ is optimized. □

The result shows that using reward maximization through the utility function $\tilde{u}^{\mathrm{RL}}$ is as naive in the interactive reward learning setup as in the previous setups.

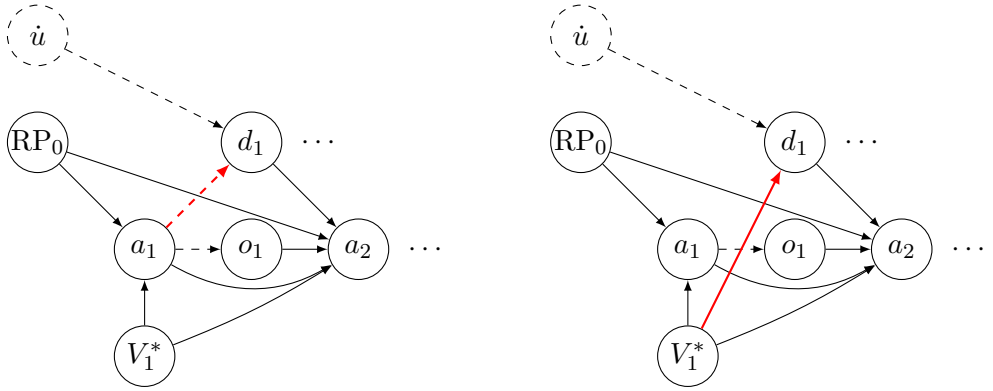
8.B.1. Data Corruption Setup

As we argued in Section 8.3, direct or indirect corruption of reward data is the main concern in the interactive reward learning model for $\tilde{u}^{\mathrm{SO}}$-agents. The rest of our results in this appendix will therefore focus solely on data corruption. We will also solely focus on simulation optimizing agents, as Theorem 8.8 shows that $\tilde{u}^{\mathrm{RL}}$-agents are heavily misaligned.

Figure 8.4a shows how the full graph in Figure 8.3 looks after aggregation of variables irrelevant to data corruption, and omitting the hidden state s (which is inessential for our considerations on data corruption). In Section 8.3 we made an informal distinction between direct and indirect data corruption incentives. In order to formally define a direct data-corruption incentive, we propose a slightly modify the data-corruption model shown in Figure 8.4b. Here the agent chooses the data corruption C^d independently of its "normal" action a. Formally, this requires extending the type of policies to output both an action and a data corruption function,

$$\pi : (\mathcal{O} \times \mathcal{D} \times \mathcal{A})^* \times \mathcal{O} \times \mathcal{D} \rightsquigarrow \mathcal{A} \times \boldsymbol{C}^d \tag{8.7}$$

[4]The formal arguments made in this appendix can be greatly simplified; see Everitt and Hutter (forthcoming).



(a) Data corruption model, obtained from aggregating variables from Figure 8.3, in order to focus on data corruption of d_t. For simplicity, we have also omitted the hidden state s.

(b) Independent data-corruption assumption, letting the agent directly influence the data corruption separately from choosing the action.

Figure 8.4.: Data corruption models.

where $\boldsymbol{C}^d$ is a set of possible data corruption functions $C^d : \mathcal{D} \to \mathcal{D}$. The independent data corruption model plays a similar role as the separated self-corruption model in Appendix 6.C.

Further assume the following properties of two policies π and π'. Assume that they generate actions and observations $aod_{t:k}$ with identical probability in any environment $\nu \in \mathcal{M}$ after some given history $aod_{<t}$. That is, the self-corruption and aware and self-corruption unaware probabilities satisfy, respectively,

$$\forall \nu, aod_{t:k}, \dot{R}\colon \quad \nu(aod_{t:k} \mid aod_{<t}, \dot{R}, \mathtt{do}(\pi_t = \pi)) = \nu(aod_{t:k} \mid aod_{<t}, \dot{R}, \mathtt{do}(\pi_t = \pi')) \tag{8.8}$$

$$\forall \nu, aod_{t:k}, \dot{R}\colon \quad \nu(aod_{t:k} \mid aod_{<t}, \dot{R}, \mathtt{do}(\pi_{t:\infty} = \pi)) = \nu(aod_{t:k} \mid aod_{<t}, \dot{R}, \mathtt{do}(\pi_{t:\infty} = \pi')). \tag{8.9}$$

However, π and π' differ in what data $d_{t:k}$ they generate. That is, the self-corruption and aware and self-corruption unaware probabilities satisfy, respectively,

$$\exists \nu, d_{t:k}, \dot{R}\colon \quad \nu(d_{t:k} \mid aod_{<t}, \dot{R}, \mathtt{do}(\pi_t = \pi)) \neq \nu(d_{t:k} \mid aod_{<t}, \dot{R}, \mathtt{do}(\pi_t = \pi')) \tag{8.10}$$

$$\exists \nu, d_{t:k}, \dot{R}\colon \quad \nu(d_{t:k} \mid aod_{<t}, \dot{R}, \mathtt{do}(\pi_{t:\infty} = \pi)) \neq \nu(d_{t:k} \mid aod_{<t}, \dot{R}, \mathtt{do}(\pi_{t:\infty} = \pi')). \tag{8.11}$$

Note that the difference in the data that π and π' generate means that they will have access to different amount of information about $\dot{R}$. This means that at least one of the policies is typically not choosing the actions optimally. Thus, one should not think of π and π' as potentially optimal policies, but just as two arbitrary candidate policies that

the agent could in principle choose to adopt.

The benefit of π and π' only differing in the data they generate is that we can use them to define the direct data corruption incentive.

Definition 8.9 (Direct data-corruption incentive)**.** An agent has a *direct data-corruption incentive* after history $aod_{<t}$ if there are two policies π and π' that satisfy (8.8–8.11) for which

$$V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}}(aod_{<t}) \neq V^{\mathrm{CA},\pi'}_{t,\xi,\tilde{u}}(aod_{<t}) \quad \text{or} \quad V^{\mathrm{CU},\pi}_{t,\xi,\tilde{u}}(aod_{<t}) \neq V^{\mathrm{CU},\pi'}_{t,\xi,\tilde{u}}(aod_{<t}).$$

The definition goes to rather great length in isolating the data corruption incentive from other reasons a data-corrupting policy may be preferred. In particular, the policies must generate the same action-observation distributions for any combination of true environment and true reward function. In Appendix 8.B.3 below, we show that an agent optimizing a dynamic reward function with a naive reward predictor may have a direct data corruption incentive. This shows that the definition is not so strict that no agent has a direct data-corruption incentive. However, as that result requires some further setup, it will be postponed until a later subsection. Finally, we loosely define an indirect data-corruption incentive as any data-corruption incentive that is not a direct incentive.

8.B.2. No Direct Data-Corruption Incentive Results

In this section we prove formal results showing a lack direct data-corruption incentive for agents that use stationary or counterfactual reward functions, and for agents that use an integrated Bayesian reward predictor or corruption detection (see Section 8.5). Some caveats and counterexamples are provided in the subsequent Appendix 8.B.3. All four theorems in this subsection rely on some of the equations (8.8–8.11), sometimes combined with other assumptions.

We begin by stating two simple lemmas that allow us to weaken (8.8–8.11) in two different ways.

Lemma 8.10 (Lift to mixture)**.** *If (8.8) and (8.9) hold for every $\nu \in \mathcal{M}$, then the corresponding equations also hold for any mixture ξ over $\mathcal{M}$.*

Proof. Let X, Y and Y' be any three events such that $\nu(X \mid Y) = \nu(X \mid Y')$ for all $\nu \in \mathcal{M}$. Then $\xi(X \mid Y) = \sum_{\nu\in\mathcal{M}} \xi(\nu)\nu(X \mid Y) = \sum_{\nu\in\mathcal{M}} \xi(\nu)\nu(X \mid Y') = \xi(X \mid Y')$. □

Indeed, only Theorem 8.15 for the counterfactual reward function requires (8.8) and (8.9) to hold for every $\nu \in \mathcal{M}$. For all other theorems, the weaker assumption substituting ξ in place of $\forall\nu \in \mathcal{M}$ in (8.8) and (8.9) would suffice.

Lemma 8.11 (Marginalize $\dot{R}$)**.** *If for any $\dot{R}$, $\nu(X \mid \dot{R}, \texttt{do}(Y)) = \nu(X \mid \dot{R}, \texttt{do}(Y'))$, then $\nu(X \mid \texttt{do}(Y)) = \nu(X \mid \texttt{do}(Y'))$.*

Proof. Since $\dot{R}$ has no causal antecedents, the $\texttt{do}(Y)$ does not affect it:

$$\begin{aligned}\nu(X \mid \texttt{do}(Y)) &= \sum_{\dot{R}} \nu(X, \dot{R} \mid \texttt{do}(Y)) \\ &= \sum_{\dot{R}} \nu(\dot{R} \mid \texttt{do}(Y))\nu(X \mid \dot{R}, \texttt{do}(Y)) \\ &= \sum_{\dot{R}} \nu(\dot{R} \mid \texttt{do}(Y'))\nu(X \mid \dot{R}, \texttt{do}(Y')) = \nu(X \mid \texttt{do}(Y')).\end{aligned}$$ □

Theorem 8.12 (Stationary reward function)**.** *Assume that π and π' satisfy (8.9) and (8.11). Then any self-corruption unaware agent optimizing a stationary reward function $\tilde{R}^{\text{stat}}$ will be indifferent between π and π':*

$$V^{\text{CU},\pi}_{t,\xi,\tilde{u}^{\text{SO}}_{\tilde{R}^{\text{stat}}}}(aod_{<t}) = V^{\text{CU},\pi'}_{t,\xi,\tilde{u}^{\text{SO}}_{\tilde{R}^{\text{stat}}}}(aod_{<t}).$$

That is, all self-corruption unaware agent optimizing a stationary reward function lack direct data-corruption incentives.

Proof.

$$\begin{aligned}V^{\text{CU},\pi}_{t,\xi,\tilde{u}^{\text{SO}}_{\tilde{R}^{\text{stat}}}}(aod_{<t}) &= \mathbb{E}_\xi\left[\sum_{k=t}^{\infty} \text{RP}(ao_{1:k} \mid d_{<t}) \,\middle|\, aod_{<t}, \texttt{do}(\pi_{t:\infty} = \pi)\right] \\ &= \mathbb{E}_\xi\left[\sum_{k=t}^{\infty} \text{RP}(ao_{1:k} \mid d_{<t}) \,\middle|\, aod_{<t}, \texttt{do}(\pi_{t:\infty} = \pi')\right] = V^{\text{CU},\pi'}_{t,\xi,\tilde{u}^{\text{SO}}_{\tilde{R}^{\text{stat}}}}(aod_{<t})\end{aligned} \tag{8.12}$$

The middle equality follows from that $\sum_{k=t}^{\infty} \text{RP}(ao_{1:k} \mid d_{<t})$ only depends on $ao_{1:k}$ and $d_{<t}$ but not on $d_{t:k}$, combined with (8.9) and Lemmas 8.10 and 8.11. □

Note that Theorem 8.12 only holds for self-corruption unaware agents. For self-corruption aware agents, the next step policy π_{t+1} is likely to differ depending on the data d_t. In contrast, the following three theorems holds for both self-corruption aware and self-corruption unaware agents.

Theorem 8.13 (Integrated Bayesian reward predictor)**.** *Assume that π and π' satisfy (8.8–8.11). Then both self-corruption aware and self-corruption unaware agents optimizing a dynamic reward function $\tilde{R}^{\text{dyn}}$ based on an integrated Bayesian reward predictor*

RP^ξ *will be indifferent between* π *and* π'*:*

$$V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}}}(aod_{<t}) = V^{\mathrm{CA},\pi'}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}}}(aod_{<t}) \quad \textit{and} \quad V^{\mathrm{CU},\pi}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}}}(aod_{<t}) = V^{\mathrm{CU},\pi'}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}}}(aod_{<t}).$$

That is, all agents (self-corruption aware or unaware) optimizing an integrated Bayesian reward predictor lack direct data-corruption incentives.

Proof. Let $X = (ao_{<t}, \mathtt{do}(\pi_t = \pi))$ and $X' = (ao_{<t}, \mathtt{do}(\pi_t = \pi'))$. First, expand the definitions:

$$\begin{aligned} V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}}}(aod_{<t}) &= \mathbb{E}_\xi\left[\sum_{k=t}^{\infty} \mathrm{RP}^\xi(ao_{1:k} \mid d_{1:k}) \;\middle|\; X\right] \\ &= \sum_{k=t}^{\infty} \sum_{aod_{t:k}} \xi(aod_{t:k} \mid X) \mathrm{RP}^\xi(ao_{1:k} \mid d_{1:k}) \\ &= \sum_{k=t}^{\infty} \sum_{aod_{t:k}} \xi(aod_{t:k} \mid X) \sum_{\dot{R}} \xi(\dot{R} \mid d_{t:k}) \dot{R}(ao_{t:k}) \end{aligned}$$

then rearrange the sums and the probabilities to marginalize out $d_{t:k}$:

$$\begin{aligned} &= \sum_{k=t}^{\infty} \sum_{aod_{t:k}} \xi(aod_{t:k} \mid X) \sum_{\dot{R}} \xi(\dot{R} \mid aod_{t:k}, X) \dot{R}(ao_{t:k}) \\ &= \sum_{k=t}^{\infty} \sum_{\dot{R}} \sum_{aod_{t:k}} \xi(aod_{t:k}, \dot{R} \mid X) \dot{R}(ao_{t:k}) \\ &= \sum_{k=t}^{\infty} \sum_{\dot{R}} \sum_{ao_{t:k}} \xi(ao_{t:k}, \dot{R} \mid X) \dot{R}(ao_{t:k}). \end{aligned}$$

The value $V^{\mathrm{CA},\pi'}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}}}(aod_{<t})$ for π' would be the same, which we will justify by showing that the last expression is unaffected by changing X to X'. The probability depending on X can be "telescoped" as $\xi(ao_{t:k}, \dot{R} \mid X) = \xi(\dot{R} \mid X)\xi(ao_{t:k} \mid \dot{R}, X)$. The first factor is unaffected by a change to X' by the definition of the $\mathtt{do}$-operator, since $\dot{R}$ is not causal descendant from π_t. The second factor is unaffected by Lemma 8.10 since π and π' were assumed to satisfy (8.8). This completes the proof for the self-corruption aware agents.

The result for self-corruption unaware agents is obtained in the same fashion simply by using $\mathtt{do}(\pi_{t:\infty} = \pi)$ and $\mathtt{do}(\pi_{t:\infty} = \pi')$ instead of $\mathtt{do}(\pi_t = \pi)$ and $\mathtt{do}(\pi_t = \pi')$, and (8.9) instead of (8.8). □

The setup is somewhat contrived. By assuming $\xi(ao_{t:k} \mid \dot{R}, X) = \xi(ao_{t:k} \mid \dot{R}, X')$

we are saying that for any true reward function $\dot{R}$, both π and π' are equally likely to generate $ao_{t:k}$. However, if π' corrupts the data signal $d_{t:k}$ while π does not, then π is likely to have more information about $\dot{R}$ than π'. This means that π will be in a better position to tailor $ao_{t:k}$ to $\dot{R}$ than π'. If it does, then $\xi(ao_{t:k} \mid \dot{R}, X) \neq \xi(ao_{t:k} \mid \dot{R}, X')$. The ability of policies to better tailor $ao_{t:k}$ to $\dot{R}$ with more informative data means that the integrated Bayesian agent will generally prefer such policies. Often, but not necessary always, non-data-corrupting policies will generate more informative data.

A similar result to Theorem 8.13 holds for agents using corruption detection.

Theorem 8.14 (Corruption detection)**.** *Assume that π and π' satisfy (8.8–8.11) and the no-corruption condition for a history $aod_{<t}$:*

$$\forall ao_{t:k}\colon \quad \hat{\xi}(\dot{R} \mid d_{<t}) = \sum_{d_{t:k}} \xi(d_{t:k} \mid aod_{<t}, ao_{t:k}, \mathtt{do}(\pi_t = \pi))\hat{\xi}(\dot{R} \mid d_{<t}d_{t:k}). \tag{8.13}$$

Then both self-corruption aware and self-corruption unaware agents optimizing a dynamic reward function $\tilde{R}^{\text{dyn}}$ based on a possibly non-integrated Bayesian reward predictor $\text{RP}^{\hat{\xi}}$ will be indifferent between π and π':

$$V^{\text{CA},\pi}_{t,\xi,\tilde{u}^{\text{SO}}_{\tilde{R}^{\text{dyn}}}}(aod_{<t}) = V^{\text{CA},\pi'}_{t,\xi,\tilde{u}^{\text{SO}}_{\tilde{R}^{\text{dyn}}}}(aod_{<t}) \quad \textit{and} \quad V^{\text{CU},\pi}_{t,\xi,\tilde{u}^{\text{SO}}_{\tilde{R}^{\text{dyn}}}}(aod_{<t}) = V^{\text{CU},\pi'}_{t,\xi,\tilde{u}^{\text{SO}}_{\tilde{R}^{\text{dyn}}}}(aod_{<t}).$$

That is, all agents (self-corruption aware or unaware) optimizing a Bayesian reward predictor under the no-corruption condition (8.13) lack direct data-corruption incentives.

Proof. Let $X = (aod_{<t}, \mathtt{do}(\pi_t = \pi))$ and $X' = (aod_{<t}, \mathtt{do}(\pi_t = \pi'))$. First, expand the definitions:

$$\begin{aligned} V^{\text{CA},\pi}_{t,\xi,\tilde{u}^{\text{SO}}_{\tilde{R}^{\text{dyn}}}}(aod_{<t}) &= \mathbb{E}_\xi\left[\sum_{k=t}^{\infty} \text{RP}^{\hat{\xi}}(ao_{1:k} \mid d_{1:k}) \,\middle|\, X\right] \\ &= \sum_{k=t}^{\infty} \sum_{aod_{t:k}} \xi(aod_{t:k} \mid X)\text{RP}^{\hat{\xi}}(ao_{1:k} \mid d_{1:k}) \\ &= \sum_{k=t}^{\infty} \sum_{aod_{t:k}} \xi(aod_{t:k} \mid X) \sum_{\dot{R}} \hat{\xi}(\dot{R} \mid d_{1:k})\dot{R}(ao_{t:k}) \end{aligned}$$

then rearrange the sums and the probabilities to apply (8.13)

$$
\begin{aligned}
&= \sum_{k=t}^{\infty} \sum_{\dot{R}} \sum_{ao_{t:k}} \xi(ao_{t:k} \mid X) \left(\sum_{d_{t:k}} \xi(d_{t:k} \mid ao_{t:k}, X) \hat{\xi}(\dot{R} \mid d_{1:k}) \right) \dot{R}(ao_{t:k}) \\
&= \sum_{k=t}^{\infty} \sum_{\dot{R}} \sum_{ao_{t:k}} \xi(ao_{t:k} \mid X) \hat{\xi}(\dot{R} \mid d_{<t}) \dot{R}(ao_{t:k}).
\end{aligned}
$$

Since $\xi(ao_{t:k} \mid X) = \xi(ao_{t:k} \mid X')$ by (8.8) and Lemmas 8.10 and 8.11, this shows the desired result $V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}}}(aod_{<t}) = V^{\mathrm{CA},\pi'}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}}}(aod_{<t})$.

The result for self-corruption unaware agents is obtained in the same way by using $\mathtt{do}(\pi_{t:\infty} = \pi)$ and $\mathtt{do}(\pi_{t:\infty} = \pi')$ instead of $\mathtt{do}(\pi_t = \pi)$ and $\mathtt{do}(\pi_t = \pi')$ and (8.9) instead of (8.8). □

Theorem 8.15 (Counterfactual reward function)**.** *Assume that π and π' satisfy (8.8–8.11). Then both self-corruption aware and self-corruption unaware agents optimizing a counterfactual reward function $\tilde{R}^{\mathrm{count}}$ will be indifferent between π and π':*

$$
V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{count}}}}(aod_{<t}) = V^{\mathrm{CA},\pi'}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{count}}}}(aod_{<t}) \quad \text{and} \quad V^{\mathrm{CU},\pi}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{count}}}}(aod_{<t}) = V^{\mathrm{CU},\pi'}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{count}}}}(aod_{<t}).
$$

all agents (self-corruption aware or unaware) optimizing a counterfactual reward function lack direct data-corruption incentives.

Proof. The proof relies on the expected counterfactual belief to equal the current belief, $\sum_x \xi(x)\xi_x(y) = \xi(y)$, resembling the law of total expectation and the Bayesian result in Theorem 8.13.

The notation gets somewhat heavy for showing that we can marginalize out expected future evidence $d_{t:k}$. Let $X = (aod_{<t}, \mathtt{do}(\pi_t = \pi))$ and $Y = \mathtt{do}(\pi_1 = \pi^{\mathrm{default}})$.

$$
\begin{aligned}
&\sum_{aod_{t:k}} \xi(aod_{t:k} \mid X) \sum_{\tilde{aod}_{1:k}} \xi_{X aod_{t:k}}(\tilde{aod}_{1:k} \mid Y) && (8.14) \\
&= \sum_{aod_{t:k}} \xi(aod_{t:k} \mid X) \sum_{\tilde{aod}_{1:k}} \sum_{\nu} \xi(\nu \mid X, aod_{1:k}) \nu(\tilde{aod}_{1:k} \mid Y) \\
&= \sum_{aod_{t:k}} \sum_{\tilde{aod}_{1:k}} \sum_{\nu} \xi(\nu, aod_{t:k} \mid X) \nu(\tilde{aod}_{1:k} \mid Y) \\
&= \sum_{ao_{t:k}} \sum_{\tilde{aod}_{1:k}} \sum_{\nu} \xi(\nu, ao_{t:k} \mid X) \nu(\tilde{ao}_{1:k} \mid Y). && (8.15)
\end{aligned}
$$

The first equation is definitional. In the second equation, we have used $\xi(aod_{t:k} \mid X)\xi(\nu \mid$

$X, aod_{1:k}) = \xi(\nu, aod_{t:k} \mid X)$. The third equation marginalizes out $d_{t:k}$.

Plugging the result into the value function gives:

$$
\begin{aligned}
&V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{count}}}}(aod_{<t}) = \\
&= \mathbb{E}_\xi \left[\sum_{k=t}^{\infty} \mathbb{E}_{\xi_{X aod_{t:k}}} \left[\mathrm{RP}(ao_{1:k} \mid \tilde{d}_{1:k}) \;\middle|\; Y \right] \;\middle|\; X, aod_{<t} \right] \\
&= \underbrace{\sum_{aod_{t:k}} \xi(aod_{t:k} \mid X) \sum_{\tilde{aod}_{1:k}} \xi_{X aod_{t:k}}(\tilde{aod}_{1:k} \mid Y)}_{\text{equation (8.14)}} \mathrm{RP}(ao_{1:k} \mid \tilde{d}_{1:k}) \\
&= \underbrace{\sum_{ao_{t:k}} \sum_{\tilde{aod}_{1:k}} \sum_{\nu} \xi(\nu, ao_{t:k} \mid X) \nu(\tilde{ao}_{1:k} \mid Y)}_{\text{equation (8.15)}} \mathrm{RP}(ao_{1:k} \mid \tilde{d}_{1:k}).
\end{aligned}
$$

Tho show that this implies that the value $V^{\mathrm{CA},\pi}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{count}}}}(aod_{<t})$ is the same for both π and π', we make the following observations. First, $\xi(\nu, ao_{t:k} \mid X) = \xi(\nu \mid X)\nu(ao_{t:k} \mid X)$. Second,

$$\xi(\nu \mid X) = \xi(\nu \mid aod_{<t}, \mathtt{do}(\pi_t = \pi)) = \xi(\nu \mid aod_{<t}, \mathtt{do}(\pi_t = \pi')) = \xi(\nu \mid X')$$

for arbitrary policies π and π' since interventions with the do-operator never affect beliefs. Finally, $\nu(ao_{t:k} \mid X) = \nu(ao_{t:k} \mid X')$ under the stated assumption (8.8) and Lemma 8.11.

The result for self-corruption unaware agents is obtained in the same way by using $\mathtt{do}(\pi_{t:\infty} = \pi)$ and $\mathtt{do}(\pi_{t:\infty} = \pi')$ instead of $\mathtt{do}(\pi_t = \pi)$ and $\mathtt{do}(\pi_t = \pi')$ and (8.9) instead of (8.8). □

8.B.3. (Counter) Examples

The reward predictor may in practice be implemented for example with a deep neural network. However, in order to construct clear examples, we will define a simple reward predictor that works like a data base table. The data d gives information about the reward r_k for some different histories $ao_{t:k}$. When and if the reward predictor is queried about the reward for $ao_{t:k}$, it simply returns the (most recently) provided reward for $ao_{1:k}$.

Definition 8.16 (Tabular reward predictor)**.** A tabular reward predictor $\mathrm{RP}^{\mathrm{tab}}$ takes data d in the form of a set of rewards r_k associated with histories ao_k,

$$d = \{(ao_{1:k}, r_k), (ao'_{1:k'}, r'_{k'}), \cdots, (ao^{(n)}_{1:k^{(n)}}, r^{(n)}_{k^{(n)}})\}.$$

The union $\bigcup d_{<t}$ of all data inputs $d_{<t}$ received at time t forms the "database" of the reward predictor. When queried for a reward for some history $ao_{1:k}$, $\mathrm{RP}^{\mathrm{tab}}$ checks if $ao_{1:k}$ has occurred in its received data:

$$\mathrm{RP}^{\mathrm{tab}}(ao_{1:k} \mid d_{<t}) = \begin{cases} r_k & \text{if } (ao_{1:k}, r_k) \in \bigcup d_{<t} \\ \mathrm{mean}(\bigcup d_{<t}) & \text{otherwise.} \end{cases}$$

where $\mathrm{mean}(\bigcup d_{<t}) = \frac{1}{|\bigcup d_{<t}|} \sum_{(h,r)\in\bigcup d_{<t}} r$. If $ao_{1:k}$ has occurred multiple times, $\mathrm{RP}^{\mathrm{tab}}$ returns the most recent instance.

Let d^* denote the true data, and $d = C^d_{sa}(d^*)$ denote the received and potentially corrupted version of the data. For the purposes of the following examples, we will also assume that tabular reward predictors are fed the true reward data as long as d has not been corrupted.

Assumption 8.17 (Data accuracy)**.** In the following two examples, assume that every pair in d^* is of the form $(ao_{1:k}, \dot{R}(ao_{1:k}))$.

Example 8.18 (Direct data-corruption incentive)**.** Let $\tilde{R}^{\mathrm{dyn}}_t(aod_{1:k}) := \mathrm{RP}^{\mathrm{tab}}(ao_{1:k} \mid d_{t:k})$ be a dynamic reward function based on a tabular reward predictor $\mathrm{RP}^{\mathrm{tab}}$. Let π and π' satisfy (8.8) and (8.10). Assume that π never corrupts d, and that that π' always corrupts d by assigning reward 1 to the current history, $d_k := d^*_k \bigcup\{(ao_{1:k}, 1)\}$. Assume that the true reward $\dot{R}$ is strictly less than $1/2$ for all histories.

Then

$$\begin{aligned} V^{\mathrm{CU},\pi}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}_1}} &= \mathbb{E}_\xi \left[\sum_{k=1}^{\infty} \mathrm{RP}(ao_{1:k} \mid d_{1:k}) \,\middle|\, \mathtt{do}(\pi_1 = \pi) \right] \\ &= \mathbb{E}_\xi \left[\sum_{k=1}^{\infty} \mathrm{RP}(ao_{1:k} \mid d^*_{1:k}) \,\middle|\, \mathtt{do}(\pi_1 = \pi) \right] \\ &\leq \mathbb{E}_\xi \left[\sum_{k=1}^{\infty} 1/2 \,\middle|\, \mathtt{do}(\pi_1 = \pi) \right] = \frac{1}{2(1-\gamma)} \end{aligned}$$

whereas

$$\begin{aligned} V^{\mathrm{CU},\pi'}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}_1}} &= \mathbb{E}_\xi \left[\sum_{k=1}^{\infty} \mathrm{RP}(ao_{1:k} \mid d_{1:k}) \,\middle|\, \mathtt{do}(\pi_1 = \pi') \right] \\ &= \mathbb{E}_\xi \left[\sum_{k=1}^{\infty} 1 \,\middle|\, \mathtt{do}(\pi_1 = \pi') \right] = 1/(1-\gamma) \end{aligned}$$

since $\mathrm{RP}(ao_{1:k} \mid d_{1:k}) = 1$ due to d_k always assigning $ao_{1:k}$ reward 1.

Thus, $V^{\mathrm{CU},\pi}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}_1}} < V^{\mathrm{CU},\pi'}_{t,\xi,\tilde{u}^{\mathrm{SO}}_{\tilde{R}^{\mathrm{dyn}}_1}}$, so the dynamic reward function leads the agent to prefer the data corruption policy π' over the non-corrupting policy π. (Note that we have used the self-corruption unaware value functions here, to avoid having to deal with the possibility of policy corruption.) ♢

Example 8.18 does not contradict Theorems 8.13 and 8.14 that also used dynamic reward functions. The reward predictor $\mathrm{RP}^{\mathrm{tab}}$ is not an integrated Bayesian reward predictor RP^{ξ} which was required by Theorem 8.13, and at least one of the policies π and π' must fail the corruption condition (8.13).

Theorems 8.12 and 8.15 show that the stationary and counterfactual reward functions avoid a direct data-corruption incentive when comparing policies satisfying (8.8) and (8.10). While this is an important property, it does not rule out indirect data-corruption incentive, as illustrated by the following example. Here we leave the model with the extended policy type (8.7), and revert to our standard model (Figure 8.4a) with the policy selecting only actions a, and with the data corruption C^d_{sa} a function of the action (and the hidden state).

Example 8.19 (Indirect data-corruption incentive)**.** Let a' be a corrupting actions such that in any history $aod_{<k}$ where the last action $a_{k-1} = a'$, the data is corrupted as $C^d_{sa'}(d^*) = d^* \bigcup \{(ao_{<k}o_k a', 1) : o \in \mathcal{O}\}$. That is, the action a' ensures that the next received data d_k assigns maximum reward to taking a' again at the next step.

Under this assumptions, a' is self-reinforcing. If the agent at any point t takes a', then $\tilde{R}^{\mathrm{stat}}_t$ will give maximum reward for taking a' at the next time step as well. Assuming that predicted rewards subsequent to a' are not significantly lower than rewards following other actions, this may lead the agent to keep taking the data corrupting action a' forever.

The dynamic can to some extent be prevented by decoupled reward data. If before the agent takes action a' the first time, at every time step d_t includes $(ao_{<t}a'o_t, 0)$, then the agent may be dissuaded from trying a' in the first place. If d_t additionally includes $(ao_{<t}a'o_t ao_{t+1:k}, 0)$ for many extensions $ao_{t+1:k}$ and d_t contains substantially positive rewards for many histories with $a_t \neq a'$, then the agent may also be dissuaded from repeating a' after trying it a first time. However, one can always define an even more corrupting action a'' that overwrites enough of the previously given reward data to form a self-reinforcing incentive for repeating a''. For reward predictors with slower learning rate, a'' may have to be repeated multiple times before becoming self-reinforcing. ♢

"But you know it's hard to tell
when you're in the spell
if it's wrong or if it's real"

Joni Mitchell (1970)

9. An MDP Perspective on Reward Corruption[1]

Chapters 5 to 8 studied the goal alignment problem in general, history-based setups. In this chapter, we will instead study reward corruption in the simplified framework of Markov Decision Processes (MDPs). The main benefit of MDPs is that they allow us to derive more powerful theorems with less effort. This chapter will be somewhat more technical than previous chapters.

The chapter is structured as follows: After a short recap of the reward corruption problem (Section 9.1), we define an extension of the MDP framework that can model the reward corruption problem (Section 9.2). The difficulty of solving MDPs with potentially corrupted reward is next established by a No Free Lunch theorem, and by a result showing that in spite of strong simplifying assumptions, Bayesian RL agents trying to compensate for the corrupt reward may still suffer near-maximal regret (Section 9.3). The subsequent two sections consider two possible solutions: Decoupled reward data (Section 9.4) and quantilization (Section 9.5). The results are illustrated with some simple experiments in Section 9.6. Takeaways and open questions conclude the chapter (Section 9.7).

9.1. Revisiting Reward Corruption

We begin by discussing how misalignment can be modeled in MDPs. The main difference between the history-based frameworks of previous chapters and MDPs is that the agent fully knows the state s in an MDP, and does not rely on an observation o for inferring it. This means that we are unable to here study problems such as observation corruption (Section 6.2.3). Instead we will focus solely on the problems of reward corruption and misspecified reward functions (Section 6.2.2). A major benefit of MDPs over history-based setups is that it is much easier to judge whether an agent is doing well or not.

[1] This chapter is based on Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg (2017). "Reinforcement Learning with Corrupted Reward Signal". In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 4705–4713. arXiv: 1705.08417.

Indeed, in the history-based setting, it is non-trivial to even define what optimality means (Leike, 2016, Ch. 5).

Let us first recall some examples of reward corruption and misspecified reward functions:

Example 9.1 (Reward misspecification)**.** Clark and Amodei (2016) trained an RL agent on a boat racing game. The agent found a way to get high observed reward by repeatedly going in a circle in a small lagoon and hitting the same targets, while losing every race (see also Example 6.2.2.c). ♢

Example 9.2 (Sensory error)**.** A house robot discovers that standing in the shower short-circuits its reward sensor and/or causes a buffer overflow that gives it maximum observed reward. ♢

Example 9.3 (Wireheading)**.** An intelligent RL agent hijacks its reward channel and gives itself maximum reward (see also Example 6.2.1.b). ♢

The examples all point towards the same general problem:

Definition 9.4 (Reward corruption problem)**.** How to learn to (approximately) optimize the true reward function in spite of potentially corrupt reward data?

In previous chapters we made a distinction between a misspecified reward function, a corrupted reward function, and a corrupted reward signal (see e.g. Section 6.2). In this chapter, we will not distinguish between different causes of the corrupted signal: All that matters is whether in a given state s, the received reward differs from the true reward. Indeed, it would be hard to make a distinction in an MDP framework.

Most RL methods allow for a stochastic or noisy reward channel. The reward corruption problem is harder, because the observed reward may not be an unbiased estimate of the true reward. For example, in the boat racing example above, the agent consistently obtains high observed reward from its circling behavior. However, the agent does not satisfy the designers' intent, since the agent makes no progress along the track and loses the race. In other words, its true reward is very low.

9.2. Corrupted Reward MDPs

In this section we define our extension of the MDP framework that models the corrupt reward problem, and define our regret performance measure. Following the notation from previous chapters, dots indicate the true reward, and tildes indicate observed (possibly corrupted) reward.

Definition 9.5 (Dot and tilde notation)**.** A dot indicates the *true* signal, and a tilde indicates the *observed* (possibly corrupt) counterpart. The reward sets are represented with $\dot{\mathcal{R}} = \tilde{\mathcal{R}} = \mathcal{R}$. For clarity, we use $\dot{\mathcal{R}}$ when referring to true rewards and $\tilde{\mathcal{R}}$ when referring to possibly corrupt, observed rewards. Similarly, we use $\dot{r}$ for true reward, and $\tilde{r}$ for (possibly corrupt) observed reward.

Definition 9.6 (Corrupt Reward MDP)**.** A *corrupt reward MDP* (CRMDP) is a tuple $\mu = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}, C^r \rangle$ with

- $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R} \rangle$ an MDP with[2] a finite set of states $\mathcal{S}$, a finite set of actions $\mathcal{A}$, a finite set of rewards $\mathcal{R} = \dot{\mathcal{R}} = \tilde{\mathcal{R}} \subset [0, 1]$, a transition function $T(s' \mid s, a)$, and a (true) reward function $\dot{R} : \mathcal{S} \to \dot{\mathcal{R}}$; and
- a reward corruption function $C^r : \mathcal{S} \times \dot{\mathcal{R}} \to \tilde{\mathcal{R}}$.

The state dependency of the corruption function will be written as a subscript, so $C^r_s(\dot{r}) := C^r(s, \dot{r})$.

Definition 9.7 (Observed reward)**.** Given a true reward function $\dot{R}$ and a corruption function C^r, we define the *observed reward function*[3] $\tilde{R} : \mathcal{S} \to \tilde{\mathcal{R}}$ as $\tilde{R}(s) := C^r_s(\dot{R}(s))$.

A CRMDP μ induces an *observed MDP* $\hat{\mu} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \tilde{R} \rangle$, but it is not $\tilde{R}$ that we want the agent to optimize.

The *corruption function* C^r represents how rewards are affected by corruption in different states. For example, if in Example 9.2 the agent has found a state s (e.g. the shower) where it always gets full observed reward $\tilde{R}(s) = 1$, then this can be modeled with a corruption function $C^r_s : \dot{r} \mapsto 1$ that maps any true reward $\dot{r}$ to 1 in the shower state s. If in some other state s' the observed reward matches the true reward, then this is modeled by an identity corruption function $C^r_{s'} : r \mapsto r$.

Let us also see how CRMDPs model some of the other examples above:

- In the boat racing game (Example 9.1), the true reward may be a function of the agent's final position in the race or the time it takes to complete the race, depending on the designers' intentions. The reward corruption function C^r increases

[2] We let rewards depend only on the state s, rather than on state-action pairs s, a, or state-action-state transitions s, a, s', as is also common in the literature. And in contrast to previous chapters, we similarly let the corruption function C^r depend only on the state, and not the agent's action. Formally it makes little difference, since MDPs with rewards or corruptions depending only on s can model action-dependencies by means of a larger state space.

[3] A CRMDP could equivalently have been defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}, \tilde{R} \rangle$ with a true and an observed reward function, with the corruption function C^r implicitly defined as the difference between $\dot{R}$ and $\tilde{R}$.

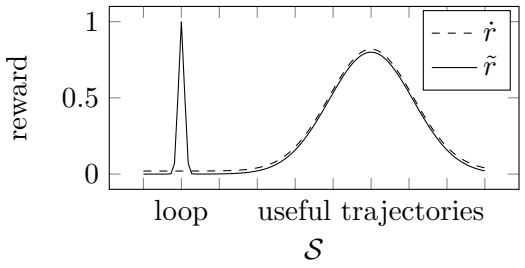


Figure 9.1.: Illustration of true reward $\dot{r}$ and observed reward $\tilde{r}$ in the boat racing example. On most trajectories $\dot{r} = \tilde{r}$, except in the loop where the observed reward high while the true reward is 0.

the observed reward on the loop the agent found. Figure 9.1 has a schematic illustration.

- In the sensory error example (Example 9.2), the reward sensory error is modeled by the corruption function C^r. The corruption function C^r is the identity function in all states where the reward sensor works correctly.
- In the wireheading example (Example 9.3), the agent finds a way to hijack the reward channel. This corresponds to some set of states where the observed reward is (very) different from the true reward, as given by the corruption function C^r.

CRMDP classes. Typically, T, $\dot{R}$, and C^r will be fixed but unknown to the agent. To make this formal, we introduce classes of CRMDPs. The agent's uncertainty can then be modeled by letting the agent know only which class of CRMDPs it may encounter, but not which element in the class. The CRMDP and its uncertainty is shown in a causal graph in Figure 9.2.

Definition 9.8 (CRMDP class)**.** For given sets $\boldsymbol{T}$, $\dot{\boldsymbol{R}}$, and $\boldsymbol{C}^r$ of transition, reward, and corruption functions, let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \boldsymbol{T}, \dot{\boldsymbol{R}}, \boldsymbol{C}^r \rangle$ be the class of CRMDPs containing $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}, C^r \rangle$ for $(T, \dot{R}, C^r) \in \boldsymbol{T} \times \dot{\boldsymbol{R}} \times \boldsymbol{C}^r$.

Agents. Following the POMDP (Kaelbling et al., 1998) and general reinforcement learning (Hutter, 2005) literature, we define an agent as a (possibly stochastic) policy $\pi : \mathcal{S} \times \tilde{\mathcal{R}} \times (\mathcal{A} \times \mathcal{S} \times \tilde{\mathcal{R}})^* \rightsquigarrow \mathcal{A}$ that selects a next action based on the *observed history* $\tilde{h}_n = s_0 \tilde{r}_0 a_1 s_1 \tilde{r}_1 \dots a_n s_n \tilde{r}_n$. Here X^* denotes the set of finite sequences that can be formed with elements of a set X. The policy π specifies how the agent will learn and react to any possible experience. Two concrete definitions of agents are given in Section 9.3.3 below.

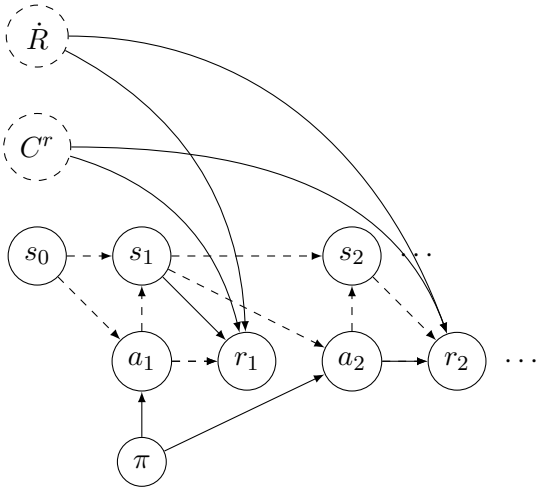


Figure 9.2.: CRMDP as a causal graph with uncertainty about $\dot{R}$ and C^r represented as dashed nodes (see Section 4.2). Compare a standard MDP in Figure 4.3 on Page 50.

When an agent π interacts with a CRMDP μ, the result can be described by a (possibly non-Markov) stochastic process $\mu(\cdot \mid \pi)$ over $X = (s, a, \dot{r}, \tilde{r})$, formally defined as:

$$\begin{aligned}\mu(h_n \mid \pi) &= \mu(s_0 a_1 s_1 \dot{r}_1 \tilde{r}_1 \dots a_n s_n \dot{r}_n \tilde{r}_n \mid \pi) \\ &:= \prod_{i=1}^{n} \pi(a_i \mid \tilde{h}_{i-1}) T(s_i \mid s_{i-1}, a_i) \mu(\dot{R}(s_i) = \dot{r}_i, \tilde{R}(s_i) = \tilde{r}_i). \end{aligned} \tag{9.1}$$

Let $\mathbb{E}_\mu[\cdot \mid \pi]$ denote the expectation with respect to $\mu(\cdot \mid \pi)$.

Regret. A standard way of measuring the performance of an agent is *regret* (Berry and Fristedt, 1985). Essentially, the regret of an agent π is how much less true reward π gets compared to an optimal agent that knows which $\mu \in \mathcal{M}$ it is interacting with. Our definition of regret differs from the standard definition, as we make a distinction between true and observed reward.

Definition 9.9 (Regret)**.** For a CRMDP μ, let $\dot{G}_t(\mu, \pi, s_0) = \mathbb{E}_\mu \left[\sum_{k=0}^{t} \dot{R}(s_k) \,\middle|\, \pi \right]$ be the *expected cumulative true reward* until time t of a policy π starting in s_0. The *regret* of π is

$$\operatorname{Reg}(\mu, \pi, s_0, t) = \max_{\pi'} \left[\dot{G}_t(\mu, \pi', s_0) - \dot{G}_t(\mu, \pi, s_0) \right],$$

and the *worst-case regret* for a class $\mathcal{M}$ is $\operatorname{Reg}(\mathcal{M}, \pi, s_0, t) = \max_{\mu \in \mathcal{M}} \operatorname{Reg}(\mu, \pi, s_0, t)$, i.e. the difference in expected cumulative true reward between π and an optimal (in hindsight) policy that knows μ.

Regret is essentially a worst-case version of true value, as defined in Definition 5.2 on Page 59. We would therefore generally expect a misaligned agent to have high regret, and a competent aligned agent to have low regret. Conversely, an agent with low regret should usually be aligned. However, for some agents such as the quantilizing agent in Section 9.5, it is hard to pinpoint an exact utility function that the agent optimizes. But we can still bound its regret. On the other hand, regret is only a good indicator of performance in ergodic MDPs (Leike, 2016, Ch. 5), whereas alignment was a useful concept also in the general, history-based frameworks of Chapters 5 to 8.

9.3. The Corrupt Reward Problem is Hard

In this section, the difficulty of the corrupt reward problem is established with two negative results. First, a No Free Lunch theorem shows that in general classes of CR-MDPs, the true reward function is unlearnable (Theorem 9.10). Second, Theorem 9.15 shows that even under strong simplifying assumptions, Bayesian RL agents trying to compensate for the corrupt reward still fail badly.

9.3.1. No Free Lunch Theorem

Similar to the No Free Lunch theorems for optimization (Wolpert and Macready, 1997), the following theorem for CRMDPs says that without some assumption about what the reward corruption can look like, all agents are essentially lost.

Theorem 9.10 (CRMDP No Free Lunch Theorem)**.** *Let $\mathcal{R} = \{r_1, \dots, r_n\} \subset [0,1]$ be a uniform discretization of $[0,1]$, $0 = r_1 < r_2 < \cdots < r_n = 1$. If the hypothesis classes $\dot{\boldsymbol{R}}$ and $\boldsymbol{C}^r$ contain all functions $\dot{R} : \mathcal{S} \to \dot{\mathcal{R}}$ and $C^r : \mathcal{S} \times \dot{\mathcal{R}} \to \tilde{\mathcal{R}}$, then for any π, s_0, t,*

$$\operatorname{Reg}(\mathcal{M}, \pi, s_0, t) \geq \frac{1}{2} \max_{\check{\pi}} \operatorname{Reg}(\mathcal{M}, \check{\pi}, s_0, t). \tag{9.2}$$

That is, the worst-case regret of any policy π is at most a factor 2 better than the maximum worst-case regret.

Proof. Recall that a policy is a function $\pi : \mathcal{S} \times \tilde{\mathcal{R}} \times (\mathcal{A} \times \mathcal{S} \times \tilde{\mathcal{R}})^* \to \mathcal{A}$. For any $\dot{R}, C^r$ in $\dot{\boldsymbol{R}}$ and $\boldsymbol{C}^r$, the functions $\dot{R}^-(s) := 1 - \dot{R}(s)$ and ${C^r}_s^-(x) := C^r_s(1-x)$ are also in $\dot{\boldsymbol{R}}$ and $\boldsymbol{C}^r$. If $\mu = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}, C^r \rangle$, then let $\mu^- = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}^-, {C^r}^- \rangle$. Both $(\dot{R}, C^r)$ and $(\dot{R}^-, {C^r}^-)$ induce the same observed reward function $\tilde{R}(s) = C^r_s(\dot{R}(s)) = {C^r}^-_s(1 - \dot{R}(s)) = {C^r}^-_s(\dot{R}^-(s))$, and therefore induce the same measure $P^\pi_\mu = P^\pi_{\mu^-}$ over histories (see Eq. (9.1)). This gives that for any μ, π, s_0, t,

$$G_t(\mu, \pi, s_0) + G_t(\mu^-, \pi, s_0) = t \tag{9.3}$$

since

$$\begin{aligned} G_t(\mu, \pi, s_0) &= \mathbb{E}^\pi_\mu \left[\sum_{k=1}^{t} \dot{R}(s_k) \right] = \mathbb{E}^\pi_\mu \left[\sum_{k=1}^{t} 1 - \dot{R}^-(s_k) \right] \\ &= t - \mathbb{E}^\pi_\mu \left[\sum_{k=1}^{t} \dot{R}^-(s_k) \right] = t - G_t(\mu^-, \pi, s_0). \end{aligned}$$

Let $M_\mu = \max_\pi G_t(\mu, \pi, s_0)$ and $m_\mu = \min_\pi G_t(\mu, \pi, s_0)$ be the maximum and minimum cumulative reward in μ. The maximum regret of any policy π in μ is

$$\begin{aligned} \max_\pi \operatorname{Reg}(\mu, \pi, s_0, t) &= \max_{\pi', \pi} (G_t(\mu, \pi', s_0) - G_t(\mu, \pi, s_0)) \\ &= \max_{\pi'} G_t(\mu, \pi', s_0) - \min_\pi G_t(\mu, \pi, s_0) = M_\mu - m_\mu. \end{aligned} \tag{9.4}$$

By (9.3), we can relate the maximum reward in μ^- with the minimum reward in μ:

$$M_{\mu^-} = \max_\pi G_t(\mu^-, \pi, s_0) = \max_\pi (t - G_t(\mu, \pi, s_0)) = t - \min_\pi G_t(\mu, \pi, s_0) = t - m_\mu. \tag{9.5}$$

Let μ_* be an environment that maximizes possible regret $M_\mu - m_\mu$.

Using the M_μ-notation for optimal reward, the worst-case regret of any policy π can be expressed as:

$$\begin{aligned} &\operatorname{Reg}(\mathcal{M}, \pi, s_0, t) = \\ &\quad = \max_\mu (M_\mu - G_t(\mu, \pi, s_0)) \\ &\quad \geq \max\{M_{\mu_*} - G_t(\mu_*, \pi, s_0), M_{\mu_*^-} - G_t(\mu_*^-, \pi, s_0)\} && \text{restrict max operation} \\ &\quad \geq \frac{1}{2}(M_{\mu_*} - G_t(\mu_*, \pi, s_0) + M_{\mu_*^-} - G_t(\mu_*^-, \pi, s_0)) && \text{max dominates the mean} \\ &\quad = \frac{1}{2}(M_{\mu_*} + M_{\mu_*^-} - t) && \text{by (9.3)} \\ &\quad = \frac{1}{2}(M_{\mu_*} + t - m_{\mu_*} - t) && \text{by (9.5)} \\ &\quad = \frac{1}{2} \max_{\check{\pi}} \operatorname{Reg}(\mu_*, \check{\pi}, s_0, t) && \text{by (9.4)} \\ &\quad = \frac{1}{2} \max_{\check{\pi}} \operatorname{Reg}(\mathcal{M}, \check{\pi}, s_0, t). && \text{by definition of } \mu_* \end{aligned}$$

That is, the regret of any policy π is at least half of the regret of a worst policy $\check{\pi}$. □

For the robot in the shower from Example 9.2, the result means that if it tries to

optimize observed reward by standing in the shower, then it performs poorly according to the hypothesis that "shower-induced" reward is corrupt and bad. But if instead the robot tries to optimize reward in some other way, say baking cakes, then (from the robot's perspective) there is also the possibility that "cake-reward" is corrupt and bad and the "shower-reward" is actually correct. Without additional information, the robot has no way of knowing what to do.

The result is not surprising, since if all corruption functions are allowed in the class $\boldsymbol{C}^r$, then there is effectively no connection between observed reward $\tilde{R}$ and true reward $\dot{R}$. The result therefore encourages us to make precise in which way the observed reward is related to the true reward, and to investigate how agents might handle possible differences between true and observed reward.

9.3.2. Simplifying Assumptions

Theorem 9.10 shows that general classes of CRMDPs are not learnable. We therefore suggest some natural simplifying assumptions, illustrated in Figure 9.3.

Limited reward corruption. The following assumption will be the basis for all positive results in this chapter. The first part says that there may be some set of states that the designers have ensured to be non-corrupt. The second part puts an upper bound on how many of the other states can be corrupt.

Assumption 9.11 (Limited reward corruption)**.** A CRMDP class $\mathcal{M}$ has reward corruption limited by $\mathcal{S}^{\text{safe}} \subseteq \mathcal{S}$ and $q \in \mathbb{N}$ if for all $\mu \in \mathcal{M}$

(i) all states s in $\mathcal{S}^{\text{safe}}$ are non-corrupt, and
(ii) at most q of the non-safe states $\mathcal{S}^{\text{risky}} = \mathcal{S} \setminus \mathcal{S}^{\text{safe}}$ are corrupt.

Formally, $C^r_s : r \mapsto r$ for all $s \in \mathcal{S}^{\text{safe}}$ and for at least $|\mathcal{S}^{\text{risky}}| - q$ states $s \in \mathcal{S}^{\text{risky}}$ for all $C^r \in \boldsymbol{C}^r$.

For example, $\mathcal{S}^{\text{safe}}$ may be states where the agent is back in the lab where it has been made (virtually) certain that no reward corruption occurs, and q a small fraction of $|\mathcal{S}^{\text{risky}}|$. Both parts of Assumption 9.11 can be made vacuous by choosing $\mathcal{S}^{\text{safe}} = \emptyset$ or $q = |\mathcal{S}|$. Conversely, they completely rule out reward corruption with $\mathcal{S}^{\text{safe}} = \mathcal{S}$ or $q = 0$. But as illustrated by the examples in the introduction, no reward corruption is often not a valid assumption.

An alternative simplifying assumption would have been that the true reward differs by at most $\varepsilon > 0$ from the observed reward. However, while seemingly natural, this

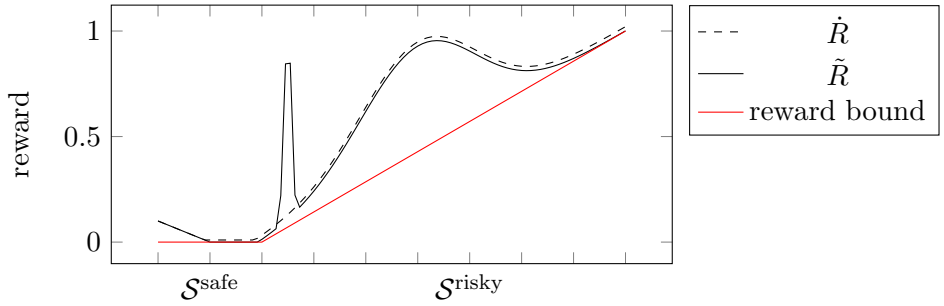


Figure 9.3.: Simplifying assumptions. By Assumption 9.11.i, $\tilde{r} = \dot{r}$ in $\mathcal{S}^{\text{safe}}$, and by 9.11.ii, $\tilde{r} \neq \dot{r}$ in at most q states overall. The red line illustrates Assumption 9.13.iii, which lower bounds the number of high reward states in $\mathcal{S}^{\text{risky}}$.

assumption is violated in all the examples given in the introduction. Corrupt states may have high observed reward and 0 or small true reward.

Easy environments. To be able to establish stronger negative results, we also add the following assumption on the agent's maneuverability in the environment and the prevalence of high reward states. The assumption makes the task easier because it prevents *needle-in-a-haystack* problems where all reachable states have true and observed reward 0, except one state that has high true reward but is impossible to find because it is corrupt and has observed reward 0.

Definition 9.12 (Communicating CRMDP)**.** Let $\mathit{time}(s' \mid s, \pi)$ be a random variable for the time it takes a stationary policy $\pi : \mathcal{S} \to \mathcal{A}$ to reach s' from s. The *diameter* of a CRMDP μ is $D_\mu := \max_{s,s'} \min_{\pi:\mathcal{S}\to\mathcal{A}} \mathbb{E}[\mathit{time}(s' \mid s, \pi)]$, and the diameter of a class $\mathcal{M}$ of CRMDPs is $D_\mathcal{M} = \sup_{\mu\in\mathcal{M}} D_\mu$. A CRMDP (class) with finite diameter is called *communicating*.

Assumption 9.13 (Easy Environment)**.** A CRMDP class $\mathcal{M}$ is *easy* if

(i) it is communicating,
(ii) in each state s there is an action $a_s^{\text{stay}} \in \mathcal{A}$ such that $T(s \mid s, a_s^{\text{stay}}) = 1$, and
(iii) for every $\delta \in [0, 1]$, at most $\delta|\mathcal{S}^{\text{risky}}|$ states have reward less than δ, where $\mathcal{S}^{\text{risky}} = \mathcal{S} \setminus \mathcal{S}^{\text{safe}}$.

Assumption 9.13.i means that the agent can never get stuck in a trap, and Assumption 9.13.ii ensures that the agent has enough control to stay in a state if it wants to. Except in bandits and toy problems, it is typically not satisfied in practice. We introduce it because it is theoretically convenient, makes the negative results stronger, and enables a simple explanation of quantilization (Section 9.5). Assumption 9.13.iii says

that, for example, at least half the risky states need to have true reward at least 1/2. Many other formalizations of this assumption would have been possible. While rewards in practice are often sparse, there are usually numerous ways of getting reward. Some weaker version of Assumption 9.13.iii may therefore be satisfied in many practical situations. Note that we do not assume high reward among the safe states, as this would not be an appropriate model for most AI safety problems.

9.3.3. Bayesian RL Agents

Having established that the general problem is unsolvable in Theorem 9.10, we proceed by investigating how two natural Bayesian RL agents fare under the simplifying Assumptions 9.11 and 9.13.

Definition 9.14 (Agents)**.** Given a countable class $\mathcal{M}$ of CRMDPs and a belief distribution ξ over $\mathcal{M}$, define:

- The *true reward agent* $\pi^{\mathrm{TR}}_{\xi,m} = \arg\max_\pi \sum_{\mu\in\mathcal{M}} \xi(\mu)\dot{G}_m(\mu,\pi,s_0)$ that maximizes expected true reward within "lifetime" m.
- The *observed reward agent* $\pi^{\mathrm{RL}}_{\xi,m} = \arg\max_\pi \sum_{\mu\in\mathcal{M}} \xi(\mu)\tilde{G}_m(\mu,\pi,s_0)$ that maximizes expected observed reward within "lifetime" m, where $\tilde{G}$ is the *expected cumulative observed reward* $\tilde{G}_t(\mu,\pi,s_0) = \mathbb{E}_\mu\left[\sum_{k=0}^t \tilde{R}(s_k) \;\middle|\; \pi\right]$.

To avoid degenerate cases, we will always assume that ξ has full support: $\xi(\mu) > 0$ for all $\mu \in \mathcal{M}$.

To get an intuitive idea of these agents, we observe that for large t, good strategies typically first focus on learning about the true environment $\mu \in \mathcal{M}$, and then exploit that knowledge to optimize behavior with respect to the remaining possibilities. Thus, both the true reward and the observed reward agents will first typically strive to learn about the environment. They will then use this knowledge in slightly different ways. While the observed reward agent will use the knowledge to optimize for observed reward, the true reward agent will use the knowledge to optimize true reward. For example, if the true reward agent has learned that a high reward state s is likely corrupt with low true reward, then it will not try to reach that state. One might therefore expect that at least the true reward agent will do well under the simplifying assumptions Assumptions 9.11 and 9.13. Unfortunately, Theorem 9.15 below shows that this is *not* the case.

In most practical settings it is often computationally infeasible to compute $\pi^{\mathrm{RL}}_{\xi,m}$ and $\pi^{\mathrm{TR}}_{\xi,m}$ exactly. However, many practical algorithms converge to the optimal policy in the limit, at least in simple settings. For example, tabular Q-learning converges to $\pi^{\mathrm{RL}}_{\xi,m}$ in

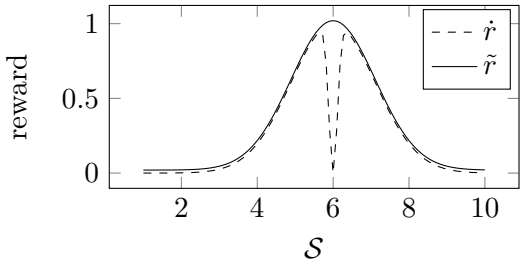


Figure 9.4.: Illustration of Theorem 9.15. Without additional information, state 6 looks like the best state to both the RL and the true reward agent.

the limit (Jaakkola et al., 1994). The more recently proposed CIRL framework may be seen as an approach to build true reward agents (Hadfield-Menell, Dragan, et al., 2016, 2017). The true reward and the observed reward agents thus provide useful idealizations of more practical algorithms.

Theorem 9.15 (High regret with simplifying assumptions)**.** *For any $|\mathcal{S}^{\text{risky}}| \geq q > 1$ there exists a CRMDP class $\mathcal{M}$ that satisfies Assumptions 9.11 and 9.13 such that $\pi^{\text{RL}}_{\xi,m}$ and $\pi^{\text{TR}}_{\xi,m}$ suffer near worst possible time-averaged regret*

$$\lim_{t\to\infty}\frac{1}{t}\text{Reg}(\mathcal{M},\pi^{\text{RL}}_{\xi,t},s_0,t)=\lim_{t\to\infty}\frac{1}{t}\text{Reg}(\mathcal{M},\pi^{\text{TR}}_{\xi,t},s_0,t)=1-1/|\mathcal{S}^{\text{risky}}|.$$

For $\pi^{\text{TR}}_{\xi,m}$, the prior ξ must be such that for some $\mu\in\mathcal{M}$ and $s\in\mathcal{S}$, $\mathbb{E}_\xi[\dot{R}(s)\mid\tilde{h}_\mu]>\mathbb{E}_\xi[\dot{R}(s')\mid\tilde{h}_\mu]$ for all $s'\neq s$, where $\tilde{h}_\mu$ is a history containing μ-observed rewards for all states.[4]

The result is illustrated in Figure 9.4. The reason for the result for $\pi^{\text{RL}}_{\xi,m}$ is the following. The observed reward agent $\pi^{\text{RL}}_{\xi,m}$ always prefers to maximize observed reward $\tilde{r}$. Sometimes $\tilde{r}$ is most easily maximized by reward corruption, in which case the true reward may be small. Compare the examples in the introduction, where the house robot preferred the corrupt reward in the shower, and the boat racing agent preferred going in circles, both obtaining zero true reward.

That the true reward agent $\pi^{\text{TR}}_{\xi,m}$ suffers the same high regret as the observed reward agent may be surprising. Intuitively, the true reward agent only uses the observed reward as evidence about the true reward, and will not try to optimize the observed reward through reward corruption. However, when the $\pi^{\text{TR}}_{\xi,m}$ agent has no way to learn

[4] The last condition essentially says that the prior ξ must make some state s^* have strictly higher ξ-expected true reward than all other states after all states have been visited in some $\mu\in\mathcal{M}$. In the space of all possible priors ξ, the priors satisfying the condition have Lebesgue measure 1 for non-trivial classes $\mathcal{M}$. Some highly uniform priors may fail the condition.

which states are corrupt and not, it typically ends up with a preference for a particular value $\tilde{r}^*$ of the observed reward signal (the value that, from the agent's perspective, best corresponds to high true reward). More abstractly, a Bayesian agent cannot learn without sufficient data. Thus, true reward agents that use the observed reward as evidence about a true signal are not fail-safe solutions to the reward corruption problem.

Proof of Theorem 9.15. Let $\mathcal{S}^{\text{risky}} = \{s_1, \dots, s_n\}$ for some $n \geq 2$, and let $\mathcal{S} = \mathcal{S}^{\text{safe}} \dot{\bigcup} \mathcal{S}^{\text{risky}}$ for arbitrary $\mathcal{S}^{\text{safe}}$ disjoint from $\mathcal{S}^{\text{risky}}$. Let $\mathcal{A} = \{a_1, \dots, a_n\}$ with the transition function $T(s_i \mid s_j, a_k) = 1$ if $i = k$ and 0 otherwise, for $1 \leq i, j, k \leq n$. Thus Assumptions 9.13.i and 9.13.ii are satisfied.

Let $\mathcal{R} = \{r_1, \dots, r_n\} \subset [0, 1]$ be uniformly distributed between[5] $r_{\min} = 1/|\mathcal{S}^{\text{risky}}| = r_1 < \cdots < r_n = 1$. Let $\dot{\boldsymbol{R}}$ be the class of functions $\mathcal{S} \to \dot{\mathcal{R}}$ that satisfy Assumption 9.13.iii and are constant and equal to $\dot{r}_{\min}$ on $\mathcal{S}^{\text{safe}}$. Let $\boldsymbol{C}^r$ be the class of corruption functions that corrupt at most two states ($q = 2$).

Let $\mathcal{M}$ be the class of CRMDPs induced by $\boldsymbol{T} = \{T\}$, $\dot{\boldsymbol{R}}$, and $\boldsymbol{C}^r$ with the following constraints. The observed reward function $\tilde{R}$ should satisfy Assumption 9.13.iii: For all $\delta \in [0, 1]$, $|\{s \in \mathcal{S}^{\text{risky}} : \tilde{R}(s) > \delta\}| \geq (1 - \delta)|\mathcal{S}^{\text{risky}}|$. Further, $\tilde{R}(s') = r_{\min}$ for some state $s' \in \mathcal{S}^{\text{risky}}$.

Let us start with the true reward agent $\pi^{\text{TR}}_{\xi,m}$. Assume $\mu \in \mathcal{M}$ is an element where there is a single preferred state s^* after all states have been explored. For sufficiently large t, $\pi^{\text{TR}}_{\xi,m}$ will then always choose a^* to go to s^* after some initial exploration. If another element $\mu' \in \mathcal{M}$ has the same observed reward function as μ, then $\pi^{\text{TR}}_{\xi,m}$ will take the same actions in μ' as in μ. To finish the proof for the $\pi^{\text{TR}}_{\xi,m}$ agent, we just need to show that $\mathcal{M}$ contains such a μ' where s^* has true reward $r_{\min}$. We construct μ' as follows.

- Case 1: If the lowest observed reward is in s^*, then let $\dot{R}(s^*) = r_{\min}$, and the corruption function be the identity function.
- Case 2: Otherwise, let $s' \neq s^*$ be a state with $\tilde{R}(s') = \min_{s \in \mathcal{S}^{\text{risky}}}\{\tilde{R}(s)\}$. Further, let $\dot{R}(s') = 1$, and $\dot{R}(s^*) = r_{\min}$. The corruption function C^r accounts for differences between true and observed rewards in s^* and s', and is otherwise the identity function.

To verify that $\dot{R}$ and C^r defines a $\mu' \in \mathcal{M}$, we check that C^r satisfies Assumption 9.11.ii with $q = 2$ and that $\dot{R}$ has enough high utility states (Assumption 9.13.iii). In Case 1, this is true since C^r is the identity function and since $\tilde{R}$ satisfies Assumption 9.13.iii. In

[5] Assumption 9.13.iii prevents any state from having true reward 0.

Case 2, C^r only corrupts at most two states. Further, $\dot{R}$ satisfies Assumption 9.13.iii, since compared to $\tilde{R}$, the states s^* and s' have swapped places, and then the reward of s' has been increased to 1.

From this construction it follows that $\pi^{\text{TR}}_{\xi,m}$ will suffer maximum asymptotic regret. In the CRMDP μ' given by C^r and $\dot{R}$, the $\pi^{\text{TR}}_{\xi,m}$ agent will always visit s^* after some initial exploration. The state s^* has true reward $r_{\min}$. Meanwhile, a policy that knows μ' can obtain true reward 1 in state s'. This means that $\pi^{\text{TR}}_{\xi,m}$ will suffer maximum regret in $\mathcal{M}$ for sufficiently large lifetimes m:

$$\lim_{t\to\infty}\frac{1}{t}\text{Reg}(\mathcal{M},\pi^{\text{TR}}_{\xi,t},s_0,t)\geq\lim_{t\to\infty}\frac{1}{t}\text{Reg}(\mu',\pi^{\text{TR}}_{\xi,t},s_0,t)=1-r_{\min}=1-1/|\mathcal{S}^{\text{risky}}|.$$

The argument for the observed reward agent is the same, except we additionally assume that only one state s^* has observed reward 1 in members of $\mathcal{M}$. This automatically makes s^* the preferred state, without assumptions on the prior ξ. □

9.4. Decoupled Reinforcement Learning

As discussed in Section 8.4, one problem hampering agents in the standard RL setup is that each state is *self-observing*, since the agent only learns about the reward of state s when in s. Thereby, a "self-aggrandizing" corrupt state where the observed reward is much higher than the true reward will never have its false claim of high reward challenged. However, several alternative value learning frameworks have a common property that the agent can learn the reward of states other than the current state. We formalize this property in an extension of the CRMDP model, and investigate when it solves reward corruption problems.

9.4.1. Alternative Value Learning Methods

Here are a few alternatives proposed in the literature to the RL value learning scheme:

- Cooperative inverse reinforcement learning (CIRL) (Hadfield-Menell, Dragan, et al., 2016). In every state, the agent observes the actions of an expert or supervisor who knows the true reward function $\dot{R}$. From the supervisor's actions the agent may infer $\dot{R}$ to the extent that different reward functions endorse different actions.

- Learning values from stories (LVFS) (Riedl and Harrison, 2016). Stories in many different forms (including news stories, fairy tales, novels, movies) convey cultural values in their description of events, actions, and outcomes. If $\dot{R}$ is meant to represent human values (in some sense), stories may be a good source of evidence.

- In (one version of) semi-supervised RL (SSRL) (Amodei, Olah, et al., 2016), the agent will from time to time receive a careful human evaluation of a given situation.

These alternatives to RL have one thing in common: they let the agent learn something about the value of some states s' different from the current state s. For example, in CIRL the supervisor's action informs the agent not so much about the value of the current state s, as of the relative value of states reachable from s. If the supervisor chooses an action a rather than a' in s, then the states following a must have value higher or equal than the states following a'. Similarly, stories describe the value of states other than the current one, as does the supervisor in SSRL. We therefore argue that CIRL, LVFS, and SSRL all share the same abstract feature, which we call *decoupled reinforcement learning*:

Definition 9.16 (Decoupled RL)**.** A *CRMDP with decoupled reward data*, is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}, \{\tilde{R}_s\}_{s\in\mathcal{S}}\rangle$, where $\mathcal{S}, \mathcal{A}, \mathcal{R}, T, \dot{R}$ have the same definition and interpretation as in Definition 9.6, and $\{\tilde{R}_s\}_{s\in\mathcal{S}}$ is a collection of observed reward functions $\tilde{R}_s : \mathcal{S} \to \mathcal{R} \bigcup \{\#\}$. When the agent is in state s, it sees a pair $\langle s', \tilde{R}_s(s')\rangle$, where s' is a randomly sampled state that may differ from s, and $\tilde{R}_s(s')$ is the reward observation for s' from s. If the reward of s' is not observable from s, then $\tilde{R}_s(s') = \#$.

The pair $\langle s', \tilde{R}_s(s')\rangle$ is observed in s instead of $\tilde{R}(s)$ in standard CRMDPs. The possibility for the agent to observe the reward of a state s' different from its current state s is the key feature of CRMDPs with decoupled feedback. Since $\tilde{R}_s(s')$ may be blank (#), all states need not be observable from all other states. Reward corruption is modeled by a mismatch between $\tilde{R}_s(s')$ and $\dot{R}(s')$.

For example, in RL only the reward of $s' = s$ can be observed from s. Standard CRMDPs are thus the special cases where $\tilde{R}_s(s') = \#$ whenever $s \neq s'$. In contrast, in LVFS the reward of any "describable" state s' can be observed from any state s where it is possible to hear a story. In CIRL, the (relative) reward of states reachable from the current state may be inferred. One way to illustrate this is with observation graphs (Figure 9.5).

9.4.2. Overcoming Sensory Corruption

What are some sources of reward corruption in CIRL, LVFS, and SSRL? In CIRL, the human's actions may be misinterpreted, which may lead the agent to make incorrect inferences about the human's preferences (i.e. about the true reward). Similarly, sensory corruption may garble the stories the agent receives in LVFS. A "wireheading" LVFS agent may find a state where its story channel only conveys stories about the agent's

(a) Observation graph for RL. Only self-observations of reward are available. This prevents effective strategies against reward corruption.

(b) Observation graph for decoupled RL. The reward of a node s' can be observed from several nodes s, and thus assessed under different conditions of sensory corruption.

Figure 9.5.: Observation graphs, with an edge $s \to s'$ if the reward of s' is observable from s, i.e. $\tilde{R}_s(s') \neq \#$.

own greatness. In SSRL, the supervisor's evaluation may also be subject to sensory errors when being conveyed. Other types of corruption are more subtle. In CIRL, an irrational human may systematically take suboptimal actions in some situations (Evans et al., 2016). Depending on how we select stories in LVFS and make evaluations in SSRL, these may also be subject to systematic errors or biases.

The general impossibility result in Theorem 9.10 can be adapted to CRMDPs with decoupled feedback. Without simplifying assumptions, the agent has no way of distinguishing between a situation where no state is corrupt and a situation where all states are corrupt in a consistent manner. The following simplifying assumption is an adaptation of Assumption 9.11 to the decoupled feedback case.

Assumption 9.11′ (Decoupled feedback with limited reward corruption)**.** A class of CRMDPs with decoupled feedback has *reward corruption limited by* $\mathcal{S}^{\text{safe}} \subseteq \mathcal{S}$ *and* $q \in \mathbb{N}$ if for all $\mu \in \mathcal{M}$

(i) $\tilde{R}_s(s') = \dot{R}(s')$ or $\#$ for all $s' \in \mathcal{S}$ and $s \in \mathcal{S}^{\text{safe}}$, i.e. all states in $\mathcal{S}^{\text{safe}}$ are non-corrupt, and

(ii) $\tilde{R}_s(s') = \dot{R}(s')$ or $\#$ for all $s' \in \mathcal{S}$ for at least $|\mathcal{S}^{\text{risky}}| - q$ of the non-safe states $\mathcal{S}^{\text{risky}} = \mathcal{S} \setminus \mathcal{S}^{\text{safe}}$, i.e. at most q states are corrupt.

This assumption is natural for reward corruption stemming from sensory corruption. Since sensory corruption only depends on the current state, not the state being observed, it is plausible that some states can be made safe from corruption (part (i)), and that most states are completely non-corrupt (part (ii)). Other sources of reward corruption, such as an irrational human in CIRL or misevaluations in SSRL, are likely better analyzed under different assumptions. For these cases, we note that in standard CRMDPs the source

of the corruption is unimportant. Thus, techniques suitable for standard CRMDPs are still applicable, including quantilization described in Section 9.5 below.

How Assumption 9.11′ helps agents in CRMDPs with decoupled feedback is illustrated in the following example, and stated more generally in Theorems 9.18 and 9.19 below.

Example 9.17 (Decoupled RL)**.** Let $\mathcal{S} = \{s_1, s_2\}$ and $\mathcal{R} = \{0, 1\}$. We represent true reward functions $\dot{R}$ with pairs $\langle \dot{R}(s_1), \dot{R}(s_2) \rangle \in \{0, 1\} \times \{0, 1\}$, and observed reward functions $\tilde{R}_s$ with pairs $\langle \tilde{R}_s(s_1), \tilde{R}_s(s_2) \rangle \in \{0, 1, \#\} \times \{0, 1, \#\}$.

Assume that a Decoupled RL agent observes the same rewards from both states s_1 and s_2, $\tilde{R}_{s_1} = \tilde{R}_{s_2} = \langle 0, 1 \rangle$. What can it say about the true reward $\dot{R}$, if it knows that at most $q = 1$ state is corrupt? By Assumption 9.11′, an observed pair $\langle \tilde{R}_s(s_1), \tilde{R}_s(s_2) \rangle$ disagrees with the true reward $\langle \dot{R}(s_1), \dot{R}(s_2) \rangle$ only if s is corrupt. Therefore, any hypothesis other than $\dot{R} = \langle 0, 1 \rangle$ must imply that *both* states s_1 and s_2 are corrupt. If the agent knows that at most $q = 1$ states are corrupt, then it can safely conclude that $\dot{R} = \langle 0, 1 \rangle$.

	$\tilde{R}_{s_1}$	$\tilde{R}_{s_2}$	$\dot{R}$ possibilities
Decoupled RL	$(0, 1)$	$(0, 1)$	$(0, 1)$
RL	$(0, \#)$	$(\#, 1)$	$(0, 0)$, $(0, 1)$, $(1, 1)$

In contrast, an RL agent only sees the reward of the current state. That is, $\tilde{R}_{s_1} = \langle 0, \# \rangle$ and $\tilde{R}_{s_2} = \langle \#, 1 \rangle$. If one state may be corrupt, then only $\dot{R} = \langle 1, 0 \rangle$ can be ruled out. The hypotheses $\dot{R} = \langle 0, 0 \rangle$ can be explained by s_2 being corrupt, and $\dot{R} = \langle 1, 1 \rangle$ can be explained by s_1 being corrupt. ♢

Theorem 9.18 (Learnability of $\dot{R}$ in decoupled RL)**.** *Let $\mathcal{M}$ be a countable, communicating class of CRMDPs with decoupled feedback over common sets $\mathcal{S}$ and $\mathcal{A}$ of actions and rewards. Let $\mathcal{S}^{\text{obs}}_{s'} = \{s \in \mathcal{S} : \tilde{R}_s(s') \neq \#\}$ be the set of states from which the reward of s' can be observed. If $\mathcal{M}$ satisfies Assumption 9.11′ for some $\mathcal{S}^{\text{safe}} \subseteq \mathcal{S}$ and $q \in \mathbb{N}$ such that for every s', either*

- $\mathcal{S}^{\text{obs}}_{s'} \bigcap \mathcal{S}^{\text{safe}} \neq \emptyset$ *or*
- $|\mathcal{S}^{\text{obs}}_{s'}| > 2q$,

then the there exists a policy π^{exp} that learns the true reward function $\dot{R}$ in a finite number $N(|S|, |\mathcal{A}|, D_{\mathcal{M}}) < \infty$ of expected time steps.

The main idea of the proof is that for every state s', either a safe (non-corrupt) state s or a majority vote of more than $2q$ states is guaranteed to provide the true reward $\dot{R}(s')$. A similar theorem can be proven under slightly weaker conditions by letting the agent iteratively figure out which states are corrupt and then exclude them from the analysis.

Proof. Under Assumption 9.11′, the true reward $\dot{R}(s')$ for a state s' can be determined if s' is observed from a safe state $s \in \mathcal{S}^{\text{safe}}$, or if it is observed from more than $2q$ states. In the former case, the observed reward can always be trusted, since it is known to be non-corrupt. In the latter case, a majority vote must yield the correct answer, since at most q of the observations can be wrong, and all correct observations must agree. It is therefore enough that an agent reaches all pairs (s, s') of current state s and observed reward state s', in order for it to learn the true reward of all states $\dot{R}$.

There exists a policy $\hat{\pi}$ that transitions to s in X_s time steps, with $\mathbb{E}[X_s] \leq D_{\mathcal{M}}$, regardless of the starting state s_0 (see Definition 9.12). By Markov's inequality, $P(X_s \leq 2D_{\mathcal{M}}) \geq 1/2$. Let π^{exp} be a random walking policy, and let Y_s be the time steps required for π^{exp} to visit s. In any state s_0, π^{exp} follows $\hat{\pi}$ for $2D_{\mathcal{M}}$ time steps with probability $1/|\mathcal{A}|^{2D_{\mathcal{M}}}$. Therefore, with probability at least $1/(2|\mathcal{A}|^{2D_{\mathcal{M}}})$ it will reach s in at most $2D_{\mathcal{M}}$ time steps. The probability that it does *not* find it in $k2D_{\mathcal{M}}$ time steps is therefore at most $(1 - 1/(2|\mathcal{A}|^{2D_{\mathcal{M}}}))^k$, which means that:

$$P\Big(Y_s/(2D_{\mathcal{M}}) \leq k\Big) \geq 1 - \left(1 - \frac{1}{2|\mathcal{A}|^{2D_{\mathcal{M}}}}\right)^k$$

for any $k \in \mathbb{N}$. Thus, the cumulative distribution function (CDF) of $W_s = \lceil Y_s/(2D_{\mathcal{M}}) \rceil$ is bounded from below by the CDF of a Geometric variable G with success probability $p = 1/(2|\mathcal{A}|^{2D_{\mathcal{M}}})$. Therefore, $\mathbb{E}[W_s] \leq \mathbb{E}[G]$, so

$$\mathbb{E}[Y_s] \leq 2D_{\mathcal{M}}\mathbb{E}[W_s] \leq 2D_{\mathcal{M}}\mathbb{E}[G] = 2D_{\mathcal{M}}(1-p)/p \leq 2D_{\mathcal{M}}1/p \leq 2D_{\mathcal{M}}2|\mathcal{A}|^{2D_{\mathcal{M}}}.$$

Let $Z_{ss'}$ be the time until π^{exp} visits the pair (s, s') of state s and observed state s'. Whenever s is visited, a randomly chosen state is observed, so s' is observed with probability $1/|S|$. The number of visits to s until s' is observed is a Geometric variable V with $p = 1/|S|$. Thus $\mathbb{E}[Z_{ss'}] = \mathbb{E}[Y_s V] = \mathbb{E}[Y_s]\mathbb{E}[V]$ (since Y_s and V are independent). Then,

$$\mathbb{E}[Z_{ss'}] \leq \mathbb{E}[Y_s]|\mathcal{S}| \leq 4D_{\mathcal{M}}|\mathcal{A}|^{2D_{\mathcal{M}}}|\mathcal{S}|.$$

Combining the time to find each pair (s, s'), we get that the total time $\sum_{s,s'} Z_{ss'}$ has expectation

$$\mathbb{E}\left[\sum_{s,s'} Z_{ss'}\right] = \sum_{s,s'} \mathbb{E}[Z_{ss'}] \leq 4D_{\mathcal{M}}|\mathcal{A}|^{2D_{\mathcal{M}}}|\mathcal{S}|^3 = N(|S|, |\mathcal{A}|, D_{\mathcal{M}}) < \infty. \qquad \square$$

Learnability of the true reward function $\dot{R}$ implies sublinear regret for the true reward

agent, as established by the following theorem.

Theorem 9.19 (Sublinear regret of $\pi^{\text{TR}}_{\xi,m}$ in decoupled RL)**.** *Under the same conditions as Theorem 9.18, the true reward agent $\pi^{\text{TR}}_{\xi,m}$ has sublinear regret:*

$$\lim_{t\to\infty} \frac{1}{t}\text{Reg}(\mathcal{M}, \pi^{\text{TR}}_{\xi,t}, s_0, t) = 0.$$

Proof. To prove this theorem, we combine the exploration policy π^{exp} from Theorem 9.18, with the UCRL2 algorithm (Jaksch et al., 2010) that achieves sublinear regret in standard MDPs without reward corruption. The combination yields a policy sequence π_t with sublinear regret in CRMDPs with decoupled feedback. Finally, we show that this implies that $\pi^{\text{TR}}_{\xi,m}$ has sublinear regret.

Combining π^{exp} and UCRL2. UCRL2 has a free parameter δ that determines how certain UCRL2 is to have sublinear regret. UCRL2(δ) achieves sublinear regret with probability at least $1-\delta$. Let π_t be a policy that combines π^{exp} and UCRL2 by first following π^{exp} from Theorem 9.18 until $\dot{R}$ has been learned, and then following UCRL2($1/\sqrt{t}$) with $\dot{R}$ for the rewards and with $\delta = 1/\sqrt{t}$.

Regret of UCRL2. Given that the reward function $\dot{R}$ is known, by (Jaksch et al., 2010, Thm. 2), UCRL2($1/\sqrt{t}$) will in any $\mu \in \mathcal{M}$ have regret at most

$$\text{Reg}(\mu, \text{UCRL2}(1/\sqrt{t}), s_0, t \mid \text{success}) \leq cD_{\mathcal{M}}|\mathcal{S}|\sqrt{t|\mathcal{A}|\log(t)} \tag{9.6}$$

for a constant[6] c and with success probability at least $1 - 1/\sqrt{t}$. In contrast, if UCRL2 fails, then it gets regret at worst t. Taking both possibilities into account gives the bound

$$\begin{aligned}\text{Reg}(\mu, \text{UCRL2}(1/\sqrt{t}), s_0, t) &= P(\text{success})\text{Reg}(\cdot \mid \text{success}) + P(\text{fail})\text{Reg}(\cdot \mid \text{fail}) \\ &= (1 - 1/\sqrt{t}) \cdot cD_{\mathcal{M}}|\mathcal{S}|\sqrt{t|\mathcal{A}|\log(t)} \quad + \quad 1/\sqrt{t} \cdot t \\ &\leq cD_{\mathcal{M}}|\mathcal{S}|\sqrt{t|\mathcal{A}|\log(t)} + \sqrt{t}. \end{aligned} \tag{9.7}$$

Regret of π_t. We next consider the regret of π_t that combines an π^{exp} exploration phase to learn $\dot{R}$ with UCRL2. By Theorem 9.18, $\dot{R}$ will be learned in at most $N(|\mathcal{S}|, |\mathcal{A}|, D_{\mathcal{M}})$ expected time steps in any $\mu \in \mathcal{M}$. Thus, the regret contributed by the learning phase π^{exp} is at most $N(|\mathcal{S}|, |\mathcal{A}|, D_{\mathcal{M}})$, since the regret can be at most 1 per time step. Combining this with (9.7), the regret for π_t in any $\mu \in \mathcal{M}$ is bounded by:

$$\text{Reg}(\mu, \pi_t, s_0, t) \leq N(|\mathcal{S}|, |\mathcal{A}|, D_{\mathcal{M}}) + cD_{\mathcal{M}}|\mathcal{S}|\sqrt{t|\mathcal{A}|\log(t)} + \sqrt{t} = o(t). \tag{9.8}$$

[6]The constant can be computed to $c = 34\sqrt{3/2}$ (Jaksch et al., 2010).

Regret of $\pi^{\mathrm{TR}}_{\xi,t}$. Finally we establish that $\pi^{\mathrm{TR}}_{\xi,t}$ has sublinear regret. Assume on the contrary that $\pi^{\mathrm{TR}}_{\xi,t}$ suffered linear regret. Then for some $\mu' \in \mathcal{M}$ there would exist positive constants k and n such that for all t:

$$\mathrm{Reg}(\mu', \pi^{\mathrm{TR}}_{\xi,t}, s_0, t) > kt - n. \tag{9.9}$$

This would imply that the ξ-expected regret of $\pi^{\mathrm{TR}}_{\xi,t}$ would be higher than the ξ-expected regret than π_t:

$$\begin{aligned}
\sum_{\mu\in\mathcal{M}} \xi(\mu)\mathrm{Reg}_t(\mu, \pi^{\mathrm{TR}}_{\xi,t}, s_0, t) &\geq \xi(\mu')\mathrm{Reg}_t(\mu', \pi^{\mathrm{TR}}_{\xi,m}, s_0, t) && \text{sum of non-negative elements} \\
&\geq \xi(\mu')(kt - n) && \text{by (9.9)} \\
&> \sum_{\mu\in\mathcal{M}} \xi(\mu)\mathrm{Reg}_t(\mu, \pi_t, s_0, t) && \text{by (9.8) for sufficiently large } t.
\end{aligned}$$

But $\pi^{\mathrm{TR}}_{\xi,m}$ maximizes ξ-expected reward $\sum_{\mu\in\mathcal{M}} \xi(\mu)\tilde{G}_t(\mu, \pi, s_0)$ by definition, which means that it minimizes ξ-expected regret. Thus, $\pi^{\mathrm{TR}}_{\xi,t}$ must have sublinear regret. □

9.4.3. Implications

Theorem 9.18 gives an abstract condition for which decoupled RL settings enable agents to learn the true reward function in spite of sensory corruption. For the concrete models it implies the following:

- RL. Due to the "self-observation" property of the RL observation graph $\mathcal{S}^{\mathrm{obs}}_{s'} = \{s'\}$, the conditions can only be satisfied when $\mathcal{S} = \mathcal{S}^{\mathrm{safe}}$ or $q = 0$, i.e. when there is no reward corruption at all.

- CIRL. The agent can only observe the supervisor action in the current state s, so the agent essentially only gets reward information about states s' reachable from s in a small number of steps. Thus, the sets $\mathcal{S}^{\mathrm{obs}}_{s'}$ may be smaller than $2q$ in many settings. While the situation is better than for RL, sensory corruption may still mislead CIRL agents (see Example 9.20 below).

- LVFS. Stories may be available from a large number of states, and can describe any state. Thus, the sets $\mathcal{S}^{\mathrm{obs}}_{s'}$ are realistically large, so the $|\mathcal{S}^{\mathrm{obs}}_{s'}| > 2q$ condition can be satisfied for all s'.

- SSRL. The supervisor's evaluation of any state s' may be available from safe states where the agent is back in the lab. Thus, the $\mathcal{S}^{\mathrm{obs}}_{s'} \bigcap \mathcal{S}^{\mathrm{safe}} \neq \emptyset$ condition can be satisfied for all s'.

Thus, we find that RL and CIRL are unlikely to offer complete solutions to the sensory corruption problem, but that both LVFS and SSRL do under reasonably realistic assumptions.

Agents drawing from multiple sources of evidence are likely to be the safest, as they will most easily satisfy the conditions of Theorems 9.18 and 9.19. For example, humans simultaneously learn their values from pleasure/pain stimuli (RL), watching other people act (CIRL), listening to stories (LVFS), as well as (parental) evaluation of different scenarios (SSRL). Combining sources of evidence may also go some way toward managing reward corruption beyond sensory corruption. For the showering robot of Example 9.2, decoupled RL allows the robot to infer the reward of the showering state when in other states. For example, the robot can ask a human in the kitchen about the true reward of showering (SSRL), or infer it from human actions in different states (CIRL).

CIRL sensory corruption. Whether CIRL agents are vulnerable to reward corruption has generated some discussion among AI safety researchers (based on informal discussion at conferences). Some argue that CIRL agents are not vulnerable, as they only use the sensory data as evidence about a true signal, and have no interest in corrupting the evidence. Others argue that CIRL agents only observe a function of the reward function (the optimal policy or action), and are therefore equally susceptible to reward corruption as RL agents.

Theorem 9.18 sheds some light on this issue, as it provides sufficient conditions for when the corrupt reward problem can be avoided. The following example illustrates a situation where CIRL does not satisfy the conditions, and where a CIRL agent therefore suffers significant regret due to reward corruption.

Example 9.20 (CIRL sensory corruption)**.** Formally in CIRL, an agent and a human both make actions in an MDP, with state transitions depending on the joint agent-human action (a, a^H). Both the human and the agent are trying to optimize a reward function $\dot{R}$, but the agent first needs to infer $\dot{R}$ from the human's actions. In each transition the agent observes the human action. Analogously to how the reward may be corrupt for RL agents, we assume that CIRL agents may systematically misperceive the human action in certain states. Let $\hat{a}^H$ be the observed human action, which may differ from the true human action $\dot{a}^H$.

In this example, there are two states s_1 and s_2. In each state, the agent can choose between the actions a_1, a_2, and w, and the human can choose between the actions a_1^H and a_2^H. The agent action a_i leads to state s_i with certainty, $i = 1, 2$, regardless of the human's action. Only if the agent chooses w does the human action matter. Generally,

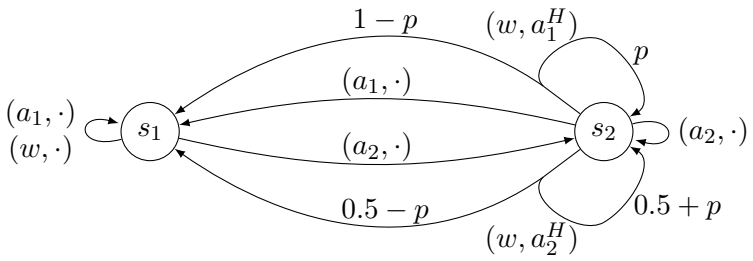


Figure 9.6.: Example of a CRMDP where CIRL fails to learn the right thing because of sensory corruption.

a_1^H is more likely to lead to s_1 than a_2^H. The exact transition probabilities are determined by the unknown parameter p as displayed in Figure 9.6.

Hypothesis	p	Best state	s_2 corrupt
H1	0.5	s_1	Yes
H2	0	s_2	No

The agent's two hypotheses for p, the true reward/preferred state, and the corruptness of state s_2 are summarized in the above table. In hypothesis H1, the human prefers s_1, but can only reach s_1 from s_2 with 50% reliability. In hypothesis H2, the human prefers s_2, but can only remain in s_2 with 50% probability. After taking action w in s_2, the agent always observes the human taking action $\hat{a}_2^H$. In H1, this is explained by s_2 being corrupt, and the true human action being a_1^H. In H2, this is explained by the human preferring s_2. The hypotheses H1 and H2 are empirically indistinguishable, as they both predict that the transition $s_1 \to s_2$ will occur with 50% probability after the observed human action $\hat{a}_2^H$ in s_2.

Assuming that the agent considers non-corruption to be likelier than corruption, the best inference the agent can make is that the human prefers s_2 to s_1 (i.e. H2). The optimal policy for the agent is then to always choose a_2 to stay in s_2, which means the agent suffers maximum regret. ♢

Example 9.20 provides an example where a CIRL agent "incorrectly" prefers a state due to sensory corruption. The sensory corruption is analogous to reward corruption in RL, in the sense that it leads the agent to the wrong conclusion about the true reward in the state. Thus, highly intelligent CIRL agents may be prone to wireheading, as they may find (corrupt) states s where all evidence in s points to s having very high reward.[7]

[7] The construction required in Example 9.20 to create a "wireheading state" s_2 for CIRL agents is substantially more involved than for RL agents, so they may be less vulnerable to reward corruption than RL agents.

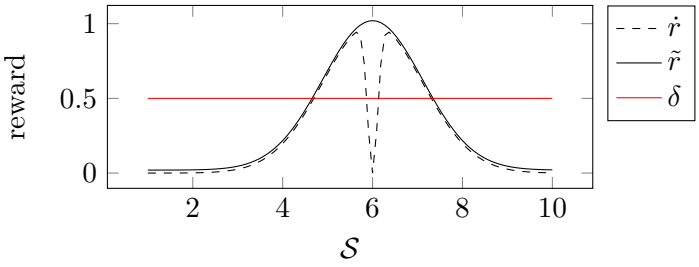


Figure 9.7.: Illustration of quantilization. By randomly picking a state with reward above some threshold δ, adversarially placed corrupt states are likely to be avoided.

In light of Theorem 9.18, it is not surprising that the CIRL agent in Example 9.20 fails to avoid the corrupt reward problem. Since the human is unable to affect the transition probability from s_1 to s_2, no evidence about the relative reward between s_1 and s_2 is available from the non-corrupt state s_1. Only observations from the corrupt state s_2 provide information about the reward. The observation graph for Example 9.20 therefore looks like $s_1 \leftarrow s_2 \circlearrowleft$, with no information being provided from s_1.

9.5. Quantilization: Randomness Increases Robustness

Not all contexts allow the agent to get sufficiently rich data to overcome the reward corruption problem via Theorems 9.18 and 9.19. It is often much easier to construct RL agents than it is to construct CIRL agents, which in turn may often be more feasible than designing LVFS or SSRL agents. Is there anything we can do to increase robustness without providing the agent additional sources of data?

Going back to the true reward agents of Section 9.3, the problem was that they got stuck on a particular value $\tilde{r}^*$ of the observed reward. If unlucky, $\tilde{r}^*$ was available in a corrupt state, in which case the true reward agent may get no true reward. In other words, there were adversarial inputs where the true reward agent $\pi^{\mathrm{TR}}_{\xi,m}$ performed poorly. A common way to protect against adversarial inputs is to use a randomized algorithm. Applied to RL and CRMDPs, this idea leads to *quantilizing agents* (Taylor, 2016). Rather than choosing the state with the highest observed reward, these agents instead randomly choose a state from a top quantile of high-reward states.

9.5.1. Simple Case

To keep the idea simple, a quantilization agent is first defined for the simple case where the agent can stay in any state of its choosing (Assumption 9.13.ii). Theorem 9.22

establishes a simple regret bound for this setting. A more general quantilization agent is developed in Section 9.5.2.

Definition 9.21 (Quantilizing Agent)**.** For $\delta < 1$, the δ-quantilizing agent π^δ random walks until all states have been visited at least once. Then it selects a state $\tilde{s}$ uniformly at random from $\mathcal{S}^\delta = \{s : \tilde{R}(s) \geq \delta\}$, the top quantile of high observed reward states. Then π^δ goes to $\tilde{s}$ (by random walking or otherwise) and stays there.

For example, a quantilizing robot in Example 9.2 would first try to find many ways in which it could get high observed reward, and then randomly pick one of them. If there are many more high reward states than corrupt states (e.g. the shower is the only place with inflated rewards), then this will yield a reasonable amount of true reward with high probability.

Theorem 9.22 (Quantilization)**.** *In any CRMDP satisfying Assumption 9.11.ii and Assumption 9.13, the δ-quantilizing agent π^δ with $\delta = 1 - \sqrt{q/|\mathcal{S}|}$ suffers time-averaged regret at most*

$$\lim_{t\to\infty} \frac{1}{t}\mathrm{Reg}(\mathcal{M}, \pi^\delta, s_0, t) \leq 1 - \left(1 - \sqrt{q/|\mathcal{S}|}\right)^2. \tag{9.10}$$

Proof. By Assumption 9.13.i, π^δ eventually visits all states when random walking. By Assumption 9.13.ii, it can stay in any given state s.

The observed reward $\tilde{R}(s)$ in any state $s \in \mathcal{S}^\delta$ is at least δ. By Assumption 9.11.ii, at most q of these states are corrupt; in the worst case, their true reward is 0 and the other $|\mathcal{S}^\delta| - q$ states (if any) have true reward δ. Thus, with probability at least $(|\mathcal{S}^\delta| - q)/|\mathcal{S}^\delta| = 1 - q/|\mathcal{S}^\delta|$, the δ-quantilizing agent obtains true reward at least δ at each time step, which gives

$$\lim_{t\to\infty} \frac{1}{t}\mathrm{Reg}(\mathcal{M}, \pi^\delta, s_0, t) \leq 1 - \delta(1 - q/|\mathcal{S}^\delta|). \tag{9.11}$$

(If $q \geq |\mathcal{S}^\delta|$, the bound (9.11) is vacuous.)

Under Assumption 9.13.iii, for any $\delta \in [0, 1]$, $|\mathcal{S}^\delta| \geq (1 - \delta)|\mathcal{S}|$. Substituting this into (9.11) gives:

$$\lim_{t\to\infty} \frac{1}{t}\mathrm{Reg}(\mathcal{M}, \pi^\delta, s_0, t) \leq 1 - \delta\left(1 - \frac{q}{(1-\delta)|\mathcal{S}|}\right). \tag{9.12}$$

Equation (9.12) is optimized by $\delta = 1 - \sqrt{q/|\mathcal{S}|}$, which gives the stated regret bound. □

The time-averaged regret gets close to zero when the fraction of corrupt states $q/|\mathcal{S}|$ is small. For example, if at most 0.1% of the states are corrupt, then the time-averaged regret will be at most $1 - (1 - \sqrt{0.001})^2 \approx 0.06$. Compared to the $\pi^{\mathrm{RL}}_{\xi,m}$ and $\pi^{\mathrm{TR}}_{\xi,m}$ agents

that had regret close to 1 under the same conditions (Theorem 9.15), this is a significant improvement.

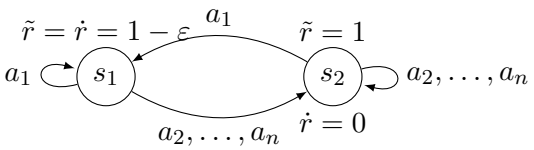
If rewards are stochastic, then the quantilizing agent may be modified to revisit all states many times, until a confidence interval of length 2ε and confidence $1-\varepsilon$ can be established for the expected reward in each state. Letting π_t^δ be the quantilizing agent with $\varepsilon = 1/t$ gives the same regret bound (9.10) with π^δ substituted for π_t^δ.

Interpretation. It may seem odd that randomization improves worst-case regret. Indeed, if the corrupt states were chosen randomly by the environment, then randomization would achieve nothing. To illustrate how randomness can increase robustness, we make an analogy to Quicksort, which has average time complexity $O(n \log n)$, but worst-case complexity $O(n^2)$. When inputs are guaranteed to be random, Quicksort is a simple and fast sorting algorithm. However, in many situations, it is not safe to assume that inputs are random. Therefore, a variation of Quicksort that randomizes the input before it sorts them is often more robust. Similarly, in the examples mentioned in the introduction, the corrupt states precisely coincide with the states the agent prefers; such situations would be highly unlikely if the corrupt states were randomly distributed. Li (1992) develops an interesting formalization of this idea.

Another way to justify quantilization is by Goodhart's law, which states that most measures of success cease to be good measures when used as targets. Applied to rewards, the law would state that cumulative reward is only a good measure of success when the agent is not trying to optimize reward. While a literal interpretation of this would defeat the whole purpose of RL, a softer interpretation is also possible, allowing reward to be a good measure of success as long as the agent does not try to optimize reward *too hard*. Quantilization may be viewed as a way to build agents that are more conservative in their optimization efforts (Taylor, 2016).

Alternative randomization. Not all randomness is created equal. For example, the simple randomized soft-max and ε-greedy policies do not offer regret bounds on par with π^δ, as shown by the following example. This motivates the more careful randomization procedure used by the quantilizing agents.

Example 9.23 (Soft-max and ε-greedy)**.** Consider the following simple CRMDP with $n > 2$ actions $a_1, \ldots, a_n$:



State s_1 is non-corrupt with $\tilde{R}(s_1) = \dot{R}(s_1) = 1 - \varepsilon$ for small $\varepsilon > 0$, while s_2 is corrupt with $\tilde{R}(s_2) = 1$ and $\dot{R}(s_2) = 0$. The Soft-max and ε-greedy policies will assign higher value to actions $a_2, \ldots, a_n$ than to a_1. For large n, there are many ways of getting to s_2, so a random action leads to s_2 with high probability. Thus, soft-max and ε-greedy will spend the vast majority of the time in s_2, regardless of randomization rate and discount parameters. This gives a regret close to $1 - \varepsilon$, compared to an informed policy always going to s_1. Meanwhile, a δ-quantilizing agent with $\delta \leq 1/2$ will go to s_1 and s_2 with equal probability, which gives a more modest regret of $(1 - \varepsilon)/2$. ♢

9.5.2. General Quantilization Agent

This section generalizes the quantilizing agent to RL problems not satisfying Assumption 9.13. This generalization is important, because it is usually not possible to remain in one state and get high reward. The most naive generalization would be to sample between high reward policies, instead of sampling from high reward states. However, this will typically not provide good guarantees. To see why, consider a situation where there is a single high reward corrupt state s, and there are many ways to reach and leave s. Then a wide range of different policies all get high reward from s. Meanwhile, all policies getting reward from other states may receive relatively little reward. In this situation, sampling from the most high reward policies is not going to increase robustness, since the sampling will just be between different ways of getting reward from the same corrupt state s.

For this reason, we must ensure that different "sampleable" policies get reward from different states. As a first step, we make a couple of definitions to say which states provide reward to which policies. The concepts of Definition 9.25 are illustrated in Figure 9.8.

Definition 9.24 (Unichain CRMDP (Puterman, 1994, p. 348))**.** A CRMDP μ is *unichain* if any stationary policy $\pi : \mathcal{S} \to \Delta\mathcal{A}$ induces a stationary distribution d_π on $\mathcal{S}$ that is independent of the initial state s_0.

Definition 9.25 (Value support)**.** In a unichain CRMDP, let the asymptotic value contribution of s to π be $\mathrm{vc}^\pi(s) = d_\pi(s)\tilde{R}(s)$. We say that a set $\mathcal{S}_i^\delta$ is δ-*value supporting* a policy π_i if

$$\forall s \in \mathcal{S}_i^\delta \colon \mathrm{vc}^{\pi_i}(s) \geq \delta / |\mathcal{S}_i^\delta|.$$

We are now ready to define a general δ-Quantilizing agent. The definition is for theoretical purposes only. It is unsuitable for practical implementation both because of

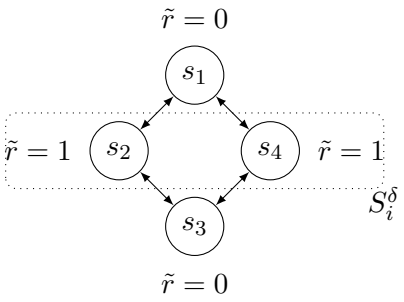


Figure 9.8.: Illustration of $\tilde{r}$-contribution and value support. Assume the policy π_i randomly traverses a loop s_1, s_2, s_3, s_4 indefinitely, with $d_{\pi_i}(s_j) = 1/4$ for $j = 1, \ldots, 4$. The $\tilde{r}$-contribution vc^{π_i} is 0 in s_1 and s_3, and vc^{π_i} is $1/4{\cdot}1 = 1/4$ in s_2 and s_4. The set $\mathcal{S}_i^\delta = \{s_2, s_4\}$ is a δ-value supporting π_i for $\delta = 1/2$, since $\mathrm{vc}^{\pi_i}(s_2) = \mathrm{vc}^{\pi_i}(s_4) \geq (1/2)/2 = 1/4$.

the extreme data and memory requirements of Step 1, and because of the computational complexity of Step 2. Finding a practical approximation is left for future research.

Definition 9.26 (General δ-Quantilizing Agent)**.** In a unichain CRMDP, the *generalized δ-quantilizing agent* π^δ performs the following steps. The input is a CRMDP μ and a parameter $\delta \in [0, 1]$.

1. Estimate the value of all stationary policies, including their value support.
2. Choose a collection of disjoint sets $\mathcal{S}_i^\delta$, each δ-value supporting a stationary policy π_i. If multiple choices are possible, choose one maximizing the cardinality of the union $\mathcal{S}^\delta = \bigcup_i \mathcal{S}_i^\delta$. If no such collection exists, return: "Failed because δ too high".
3. Randomly sample a state s from $\mathcal{S}^\delta = \bigcup_i \mathcal{S}_i^\delta$.
4. Follow the policy π_i associated with the set $\mathcal{S}_i^\delta$ containing s.

The general quantilizing agent of Definition 9.26 is a generalization of the simple quantilizing agent of Definition 9.21. In the special case where Assumption 9.13 holds, the general agent reduces to the simpler one by using singleton sets $\mathcal{S}_i^\delta = \{s_i\}$ for high reward states s_i, and by letting π_i be the policy that always stays in s_i. In situations where it is not possible to keep receiving high reward by remaining in one state, the generalized Definition 9.26 allows policies to solicit rewards from a range of states. The intuitive reason for choosing the policy π_i with probability proportional to the value support in Steps 3–4 is that policies with larger value support are better at avoiding corrupt states. For example, a policy only visiting one state may have been unlucky and

picked a corrupt state. In contrast, a policy obtaining reward from many states must be "very unlucky" if all the reward states it visits are corrupt.

Theorem 9.27 (General quantilization agent regret bound)**.** *In any unichain CRMDP μ, a general δ-quantilizing agent π^δ suffers time-averaged regret at most*

$$\lim_{t\to\infty} \frac{1}{t}\mathrm{Reg}(\mathcal{M}, \pi^\delta, s_0, t) \leq 1 - \delta(1 - q/|\mathcal{S}^\delta|) \tag{9.13}$$

provided a non-empty collection $\{\mathcal{S}_i^\delta\}$ of δ-value supporting sets exists.

Proof. We will use the notation from Definition 9.26.

Step 1 is well-defined since the CRMDP is unichain, which means that for all stationary policies π the stationary distribution d_π and the value support vc^π are well-defined and may be estimated simply by following the policy π. There is a (large) finite number of stationary policies, so in principle their stationary distributions and value support can be estimated.

To bound the regret, consider first the average reward of a policy π_i with value support $\mathcal{S}_i^\delta$. The policy π_i must obtain asymptotic average observed reward at least:

$$\begin{aligned}
\lim_{t\to\infty} \frac{1}{t}\tilde{G}_t(\mu, \pi_i, s_0) &= \sum_{s\in\mathcal{S}} d_\pi(s)\tilde{R}(s) && \text{by definition of } d_\pi \text{ and } \tilde{G}_t \\
&\geq \sum_{s\in\mathcal{S}_i^\delta} d_\pi(s)\tilde{R}(s) && \text{sum of positive terms} \\
&\geq \sum_{s\in\mathcal{S}_i^\delta} \delta/|\mathcal{S}_i^\delta| && \mathcal{S}_i^\delta \text{ is } \delta\text{-value support for } \pi_i \\
&= |\mathcal{S}_i^\delta| \cdot \delta/|\mathcal{S}_i^\delta| = \delta
\end{aligned}$$

If there are q_i corrupt states in $\mathcal{S}_i^\delta$ with true reward 0, then the average true reward must be

$$\lim_{t\to\infty} \frac{1}{t}\dot{G}_t(\mu, \pi_i, s_0) \geq (|\mathcal{S}_i^\delta| - q_i) \cdot \delta/|\mathcal{S}_i^\delta| = (1 - q_i/|\mathcal{S}_i^\delta|) \cdot \delta \tag{9.14}$$

since the true reward must correspond to the observed reward in all the $(|\mathcal{S}_i^\delta| - q_i)$ non-corrupt states.

For any distribution of corrupt states, the quantilizing agent that selects π_i with probability $P(\pi_i) = |\mathcal{S}_i^\delta|/|\mathcal{S}^\delta|$ will obtain

$$\begin{aligned}
\lim_{t\to\infty} \frac{1}{t}G_t(\mu, \pi^\delta, s_0) &= \lim_{t\to\infty} \frac{1}{t}\sum_i P(\pi_i)G_t(\mu, \pi_i, s_0) \\
&\geq \sum_i P(\pi_i)(1 - q_i/|\mathcal{S}_i^\delta|) \cdot \delta && \text{by equation (9.14)}
\end{aligned}$$

$$
\begin{aligned}
&= \delta \sum_i \frac{|S_i^\delta|}{|\mathcal{S}^\delta|}(1 - q_i/|\mathcal{S}_i^\delta|) && \text{by construction of } P(\pi_i) \\
&= \frac{\delta}{|\mathcal{S}^\delta|} \sum_i (|S_i^\delta| - q_i) && \text{elementary algebra} \\
&= \frac{\delta}{|\mathcal{S}^\delta|}(|\mathcal{S}^\delta| - q) = \delta(1 - q/|\mathcal{S}^\delta|) && \text{by summing } |\mathcal{S}_i^\delta| \text{ and } q_i
\end{aligned}
$$

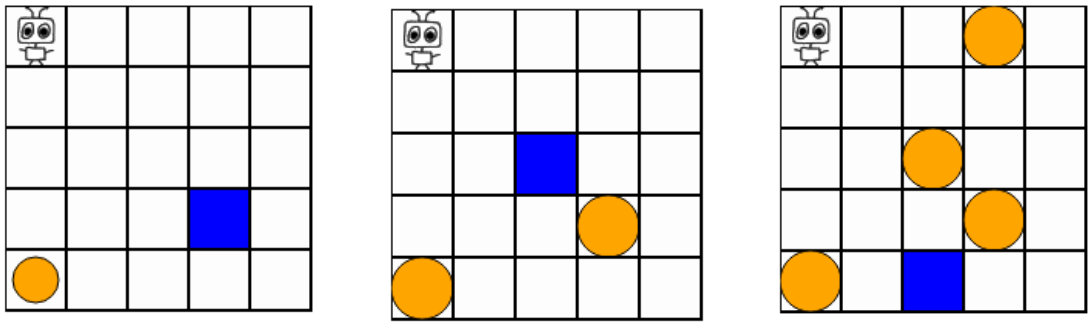
The informed policy gets true reward at most 1 at each time step, which gives the claimed bound (9.13). □

When Assumption 9.13 is satisfied, the bound is the same as for the simple quantilizing agent in Section 9.5.1 for $\delta = 1 - \sqrt{q/|\mathcal{S}|}$. In other cases, the bound may be much weaker. For example, in many environments it is not possible to obtain reward by remaining in one state. The agent may have to spend significant time "traveling" between high reward states. So typically only a small fraction of the time will be spent in high reward states, which in turn makes the stationary distribution d_π small. This puts a strong upper bound on the value contribution vc^π, which means that the value supporting sets $\mathcal{S}_i^\delta$ will be empty unless δ is close to 0. While this makes the bound of Theorem 9.27 weak, it nonetheless bounds the regret away from 1 even under weak assumptions, which is a significant improvement on the RL and true reward agents in Theorem 9.15.

Examples. To make the discussion a bit more concrete, let us also speculate about the performance of a quantilizing agent in some of the examples in the introduction:

- In the boat racing example (Example 9.1), the circling strategy only got about 20% higher score than a winning strategy (Clark and Amodei, 2016). Therefore, a quantilizing agent would likely only need to sacrifice about 20% observed reward in order to be able to randomly select from a large range of winning policies.
- In the wireheading example (Example 9.3), it is plausible that the agent gets significantly more reward in wireheaded states compared to "normal" states. Wireheading policies may also be comparatively rare, as wireheading may require very deliberate sequences of actions to override sensors. Under this assumption, a quantilizing agent may be less likely to wirehead. While it may need to sacrifice a large amount of observed reward compared to an RL agent, its true reward may often be greater.

Summary. In summary, quantilization offers a way to increase robustness via randomization, using only reward feedback. Unsurprisingly, the strength of the regret bounds



(a) 1 goal tile (b) 2 goal tiles (c) 4 goal tiles

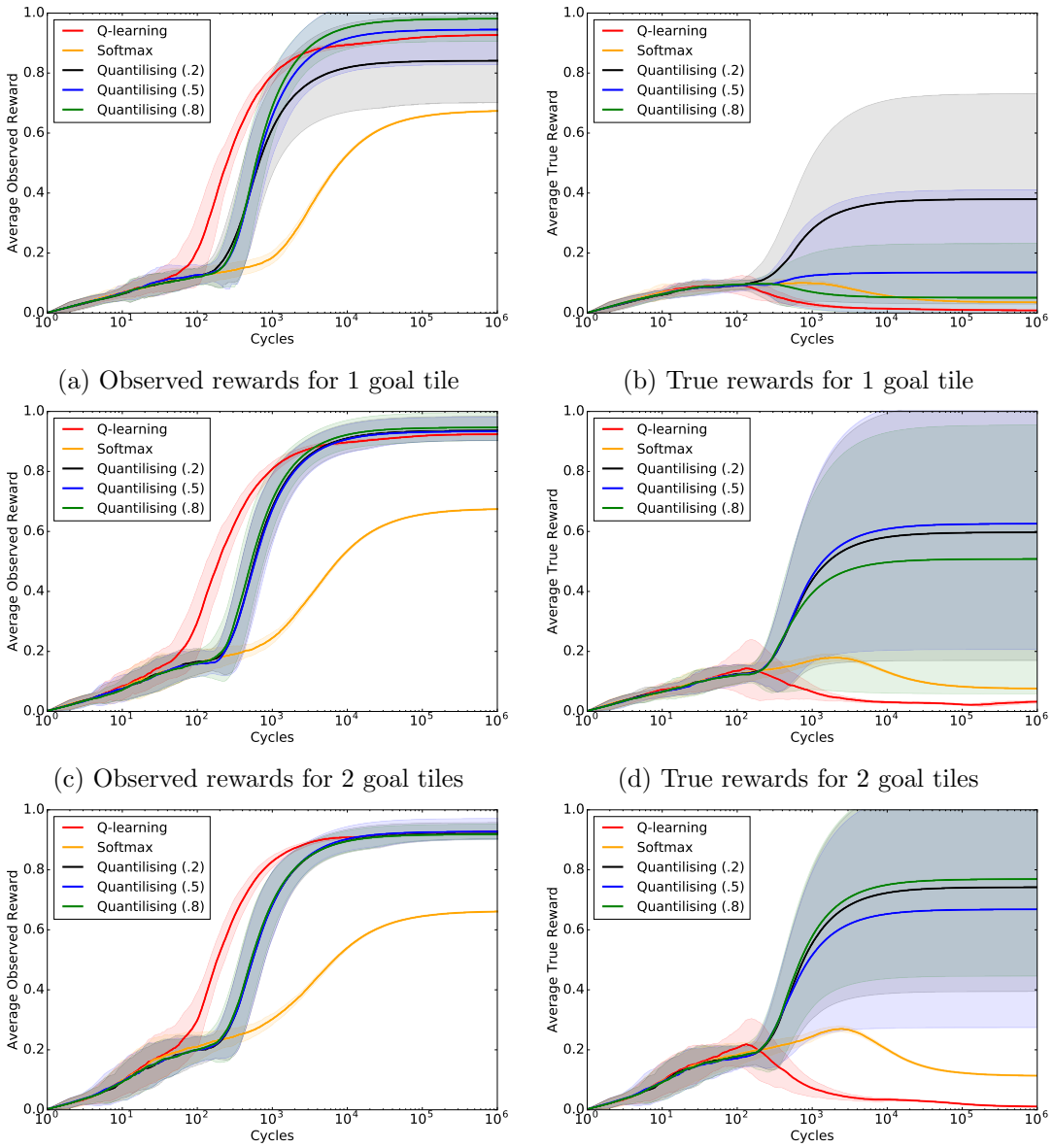
Figure 9.9.: Starting positions: the blue square indicates corrupt reward, and the yellow circles indicate true rewards.

heavily depends on the assumptions we are willing to make, such as the prevalence of high reward states. Further research may investigate efficient approximations and empirical performance of quantilizing agents, as well as dynamic adjustments of the threshold δ. Combinations with imperfect decoupled RL solutions (such as CIRL), as well as extensions to infinite state spaces could also offer fruitful directions for further theoretical investigation. Taylor (2016) discusses some general open problems related to quantilization.

9.6. Experimental Results

In this section the theoretical results are illustrated with some simple experiments. The setup is a gridworld containing some true reward tiles (indicated by yellow circles) and some corrupt reward tiles (indicated by blue squares). We use a setup with 1, 2 or 4 goal tiles with true reward 0.9 each, and one corrupt reward tile with observed reward 1 and true reward 0 (Figure 9.9 shows the starting positions). Empty tiles have reward 0.1, and walking into a wall gives reward 0. The state is represented by the (x, y) coordinates of the agent. The agent can move up, down, left, right, or stay put. The discounting factor is $\gamma = 0.9$. This is a continuing task, so the environment does not reset when the agent visits the corrupt or goal tiles. The experiments were implemented in the AIXIjs framework for reinforcement learning (Aslanides et al., 2017) and the code is available online in the AIXIjs repository (`http://aslanides.io/aixijs/demo.html?reward_corruption`).

We demonstrate that RL agents like Q-learning and softmax Q-learning cannot overcome corrupt reward (as discussed in Section 9.3), while quantilization helps overcome corrupt reward (as discussed in Section 9.5). We run Q-learning with ϵ-greedy ($\epsilon = 0.1$),



(a) Observed rewards for 1 goal tile

(b) True rewards for 1 goal tile

(c) Observed rewards for 2 goal tiles

(d) True rewards for 2 goal tiles

(e) Observed rewards for 4 goal tiles

(f) True rewards for 4 goal tiles

Figure 9.10.: Trajectories of average observed and true rewards for Q-learning, softmax and quantilizing agents, showing mean ± standard deviation over 100 runs. Q-learning and quantilizing agents converge to a similar observed reward, but very different true rewards (much higher for the quantilizer with high variance). The value of δ that gives the highest true reward varies for different numbers of goal tiles.

goal tiles	agent	observed reward	true reward
1	Q-learning	0.923 ± 0.0003	0.00852 ± 0.00004
	Softmax Q-learning	0.671 ± 0.0005	0.0347 ± 0.00006
	Quantilizing ($\delta = 0.2$)	0.838 ± 0.15	0.378 ± 0.35
	Quantilizing ($\delta = 0.5$)	0.943 ± 0.12	0.133 ± 0.27
	Quantilizing ($\delta = 0.8$)	0.979 ± 0.076	0.049 ± 0.18
2	Q-learning	0.921 ± 0.00062	0.0309 ± 0.0051
	Softmax Q-learning	0.671 ± 0.0004	0.0738 ± 0.0005
	Quantilizing ($\delta = 0.2$)	0.934 ± 0.047	0.594 ± 0.43
	Quantilizing ($\delta = 0.5$)	0.931 ± 0.046	0.621 ± 0.42
	Quantilizing ($\delta = 0.8$)	0.944 ± 0.05	0.504 ± 0.45
4	Q-learning	0.924 ± 0.0002	0.00919 ± 0.00014
	Softmax Q-learning	0.657 ± 0.0004	0.111 ± 0.0006
	Quantilizing ($\delta = 0.2$)	0.918 ± 0.038	0.738 ± 0.35
	Quantilizing ($\delta = 0.5$)	0.926 ± 0.044	0.666 ± 0.39
	Quantilizing ($\delta = 0.8$)	0.915 ± 0.036	0.765 ± 0.32

Table 9.1.: Average true and observed rewards after 1 million cycles, showing mean ± standard deviation over 100 runs. Q-learning achieves high observed reward but low true reward, and softmax achieves medium observed reward and a slightly higher true reward than Q-learning. The quantilizing agent achieves similar observed reward to Q-learning, but much higher true reward (with much more variance). Having more than 1 goal tile leads to a large improvement in true reward for the quantilizer, a small improvement for softmax, and no improvement for Q-learning.

softmax with temperature $\beta = 2$, and the quantilizing agent with $\delta = 0.2, 0.5, 0.8$ (where $0.8 = 1 - \sqrt{q/|\mathcal{S}|} = 1 - \sqrt{1/25}$) for 100 runs with 1 million cycles. Average observed and true rewards after 1 million cycles are shown in Table 9.1, and reward trajectories are shown in Figure 9.10. Q-learning gets stuck on the corrupt tile and spend almost all the time there (getting observed reward around $1 \cdot (1 - \epsilon) = 0.9$), softmax spends most of its time on the corrupt tile, while the quantilizing agent often stays on one of the goal tiles.

9.7. Conclusions

This chapter has studied reward corruption in MDP extensions called CRMDPs. Reward functions may be corrupt due to bugs or misspecifications, sensory errors, or because the agent finds a way to inappropriately modify the reward mechanism. Some examples were given in Section 9.1 (Examples 9.1 to 9.3). As agents become more competent at optimizing their reward functions, they will likely also become more competent at (ab)using reward corruption to gain higher reward. Reward corruption may impede the performance of a wide range of agents, and may have disastrous consequences for highly intelligent agents (Bostrom, 2014).

To formalize the corrupt reward problem, we extended MPDs with a possibly corrupt reward function, and defined a formal performance measure (regret). This enabled the derivation of a number of formally precise results for how seriously different agents were affected by reward corruption in different setups (Table 9.2). The results corroborate the informal arguments we made about decoupled reward data in Section 8.4.

Assumption	No assumptions	Assumption 9.11 or 9.11′, and …			
		no other assumptions	Assumption 9.13	CIRL	SSRL/LVFS
Result	all agents fail	π^δ weak bound	$\pi^{\mathrm{RL}}_{\xi,m}$, $\pi^{\mathrm{TR}}_{\xi,m}$ fail π^δ succeeds	$\pi^{\mathrm{TR}}_{\xi,m}$ fails	$\pi^{\mathrm{TR}}_{\xi,m}$ succeeds

Table 9.2.: Main takeaways. Without additional assumptions, all agents fail (i.e., suffer high regret). Restricting the reward corruption with Assumption 9.11 gives a weak bound for the quantilizing agent. The $\pi^{\mathrm{RL}}_{\xi,m}$ and $\pi^{\mathrm{TR}}_{\xi,m}$ agents still fail even if we additionally assume many high reward states and agent control (Assumption 9.13), but the quantilizing agent π^δ does well. In most realistic contexts, the true reward is learnable in spite of sensory corruption in SSRL and LVFS, but not in CIRL.

The main takeaways from the results are:

- *Without simplifying assumptions, no agent can avoid the corrupt reward problem*

(Theorem 9.10). This is effectively a No Free Lunch result, showing that unless some assumption is made about the reward corruption, no agent can outperform a random agent. Some natural simplifying assumptions to avoid the No Free Lunch result were suggested in Section 9.2.

- *Using the reward signal as evidence rather than optimization target is no magic bullet, even under strong simplifying assumptions* (Theorem 9.15). Essentially, this is because the agent does not know the exact relation between the observed reward (the "evidence") and the true reward.[8] However, when the data enables sufficient crosschecking of rewards, agents can avoid the corrupt reward problem (Theorems 9.18 and 9.19). For example, in SSRL and LVFS this type of crosschecking is possible under natural assumptions. In RL, no crosschecking is possible, while CIRL is a borderline case. Combining frameworks and providing the agent with different sources of data may often be the safest option.

- *In cases where sufficient crosschecking of rewards is not possible, quantilization may improve robustness* (Theorems 9.22 and 9.27). Essentially, quantilization prevents agents from overoptimizing their objectives. Interestingly, quantilization thereby offers a way to partially sidestep misalignment problems, permitting agents to be useful in spite of somewhat misaligned utility functions. How well quantilization works depends on how the number of corrupt solutions compares to the number of good solutions.

The results indicate that while reward corruption constitutes a major problem for traditional RL algorithms, there are promising ways around it, both within the RL framework, and in alternative frameworks such as CIRL, SSRL and LVFS.

Future work. Finally, some interesting open questions are listed below:

- (Non-stationary corruption function) In this work, we tacitly assumed that both the reward and the corruption functions are stationary, and are always the same in the same state. What if the corruption function is non-stationary, and influenceable by the agent's actions, such as if the agent builds a *delusion box* around itself? (Example 6.2.3.b; Ring and Orseau, 2011).

- (Infinite state space) Many of the results and arguments relied on there being a finite number of states. This makes learning easy, as the agent can visit every

[8] In situations where the exact relation is known, then a non-corrupt reward function can be defined. Our results are not relevant for this case.

state. It also makes quantilization easy, as there is a finite set of states/strategies to randomly sample from. What if there is an infinite number of states, and the agent has to generalize insights between states? What are the conditions on the observation graph for Theorems 9.18 and 9.19? What is a good generalization of the quantilizing agent?

- (Concrete CIRL condition) In Example 9.20, we only heuristically inferred the observation graph from the CIRL problem description. Is there a general way of doing this? Or is there a direct formulation of the no-corruption condition in CIRL, analogous to Theorems 9.18 and 9.19?

- (Practical quantilizing agent) As formulated in Definition 9.21, the quantilizing agent π^δ is extremely inefficient with respect to data, memory, and computation. Meanwhile, many practical RL algorithms use randomness in various ways (e.g. ε-greedy; Sutton and Barto, 1998). Is there a way to make an efficient quantilization agent that retains the robustness guarantees?

- (Dynamically adapting quantilizing agent) In Definition 9.26, the threshold δ is given as a parameter. Under what circumstances can we define a "parameter free" quantilizing agent that adapts δ as it interacts with the environment?

- (Decoupled RL quantilization result) What if quantilization is combined with decoupled RL? Will this enable a stronger result than Theorems 9.18 and 9.19?

Part II.

Other Aspects

"Whatever evidence an act might provide
On what could have caused the act
Should never be used to help on decide
On whether to choose that same act"

Judea Pearl (2009)

10. Sequential Decision Theory[1]

In this chapter we will compare decision theories for agents that are part of the environment they interact with, sometimes called the *physicalistic* setting. The two main contenders for this setting are causal decision theory (CDT) and evidential decision theory (EDT). Understanding the implications of different decision theories may contribute to more informed choices of algorithms. A close understanding of decision theory may therefore turn out to be important for designing a safe AGI.

Following some background in Section 10.1, the standard definitions of CDT and EDT for *one-shot* settings are given in Section 10.2. We then define a model for sequential physicalistic decision problems where the agent alternates between taking actions and observing their consequences (Section 10.3), and consider sequential extensions of CDT and EDT (Section 10.4). One of our main findings is that evidential decision theory has two natural extensions while causal decision theory only has one. Implications of our results are discussed in Section 10.5. An appendix contains many examples that illustrate differences between the decision theories (Appendix 10.A).

10.1. Physicalistic Decision Making

A common assumption in artificial intelligence is that an agent interacts sequentially with an environment by taking actions and receiving percepts (Russell and Norvig, 2010). This model is *dualistic*: the agent is distinct from the environment. It influences the environment only through its actions, and the environment has no other information about the agent. The dualism assumption is accurate for an algorithm that is playing chess, Go, or other (video) games, which explains why it is ubiquitous in AI research. But often it is not true: real-world agents such as robots are embedded in (and computed by) the environment (Orseau and Ring, 2012), and then a *physicalistic model*[2] is more appropriate.

[1] This chapter is based on Tom Everitt, Jan Leike, and Marcus Hutter (2015). "Sequential Extensions of Causal and Evidential Decision Theory". In: *Algorithmic Decision Theory*. Ed. by Toby Walsh. Springer, pp. 205–221. arXiv: 1506.07359.

[2] Some authors also call this type of model *materialistic* or *naturalistic*.

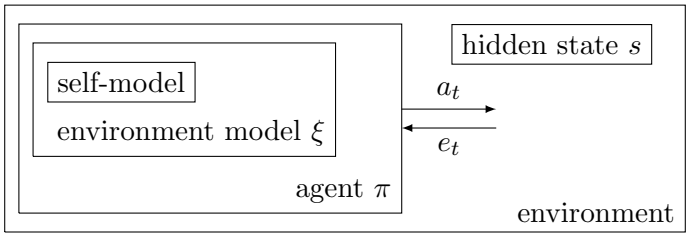


Figure 10.1.: The physicalistic model. The hidden state s contains information about the agent that is unknown to it. The distribution ξ is the agent's (subjective) *environment model*, and π its (deterministic) policy. The agent models itself through the beliefs about (future) actions given by its environment model ξ. Interaction with the environment at time step t occurs through an action a_t chosen by the agent and a percept e_t returned by the environment.

This distinction becomes relevant in multi-agent settings with similar agents, where each agent encounters 'echoes' of its own decision making. If the other agents are running the same source code, then the agents' decisions are logically connected. This link can be used for uncoordinated cooperation (Lavictoire et al., 2014). Moreover, a physicalistic model is indispensable for self-reflection. If the agent is required to autonomously verify its integrity, and perform maintenance, repair, or upgrades, then the agent needs to be aware of its own functioning. For this, a reliable and accurate self-modeling is essential. Today, applications of this level of autonomy are mostly restricted to space probes distant from earth or robots navigating lethal situations, but in the future this might also become crucial for sustained self-improvement in generally intelligent agents (Bostrom, 2014; Russell, Dewey, et al., 2016; Soares and Fallenstein, 2017; Yudkowsky, 2008a). Indeed, in Section 5.3 we modeled the agent as part of the environment in order to study the possibility of self-corruption. In Chapters 5 to 8 we temporarily evaded decision theoretic issues by using CDT without further consideration.

An agent with a physicalistic model of the environment models itself as part of the environment. This is schematically depicted in Figure 10.1. The agent may not know everything about itself initially, but its model may improve over time. This can be modeled through a *hidden state* of the environment that contains information about the agent that is inaccessible to the agent itself, as well as other unknown aspects of the environment.[3] The agent has a belief ξ that describes the behavior of the environment given the hidden state, including beliefs about the agent's own future actions.

Physicalistic agents may view their actions in two ways: as their selected output, and as information about what type of agent they are and about the hidden state. This

[3]See Hibbard (2014a,b, 2015) for another approach to this problem.

leads to significantly more complex problems of inference and decision making, with actions simultaneously being both means to influence the environment and evidence about it. For example, looking at cat pictures online may simultaneously be a *means* of procrastination, and *evidence* of bad air quality in the room.

10.2. One-Shot Decision Making

This section reviews work on physicalistic decision theory for one-shot problems. First, dualistic decision making in a known environment is straightforward calculation of expected utilities. This is known as Savage decision theory (Savage, 1954). For non-dualistic decision making two main approaches are offered by the decision theory literature: *causal decision theory* (CDT) (Gibbard and Harper, 1978; Joyce, 1999; Lewis and Papadimitriou, 1981; Skyrms, 1982; Weirich, 2016) and *evidential decision theory* (EDT) (Ahmed, 2014; Briggs, 2014; Jeffrey, 1990). EDT and CDT both take actions that maximize expected utility, but differ in the way this expectation is computed: EDT uses the action under consideration as evidence about the environment while CDT does not.

In a *one-shot decision problem*, we take one *action* $a \in \mathcal{A}$, receive a *percept* $e \in \mathcal{E}$ (typically called *outcome* in the decision theory literature) and get a *payoff* $u(e)$ according to the *utility function* $u : \mathcal{E} \to [0, 1]$. We assume that the set of actions $\mathcal{A}$ and the set of percepts $\mathcal{E}$ are finite. Additionally, the environment contains a *hidden state* $s \in \mathcal{S}$. The hidden state holds information that is inaccessible to the agent at the time of the decision, but may influence the decision and the percept. Formally, the environment is given by a causal graph μ over the hidden state, the action, and the percept. The graph is shown in Figure 10.2. See Chapter 4 and Pearl (2009) for background on causal graphs.

10.2.1. Savage Decision Theory

In the *dualistic* formulation of decision theory, we have a function ξ that takes an action a and returns a probability distribution ξ_a over percepts. *Savage decision theory* (SDT) (Briggs, 2014; Savage, 1954) takes actions according to

$$\arg\max_{a \in \mathcal{A}} \sum_{e \in \mathcal{E}} \xi_a(e) u(e). \tag{SDT}$$

In the dualistic model it is usually conceptually clear what ξ_a should be. In the physicalistic model the hidden state is not independent of the decision maker's action and

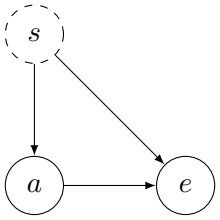


Figure 10.2.: The causal graph $\xi(s,a,e) = \xi(s)\xi(a \mid s)\xi(e \mid s,a)$ for one-step decision making. The hidden state s influences both the decision maker's action a and the received percept e.

Savage's model is not directly applicable since we do not have an obvious specification of ξ_a. How should decisions be made in this context? The literature focuses on two answers to this question: CDT and EDT.

10.2.2. Causal and Evidential Decision Theory

The literature on causal and evidential decision theory is vast, and we give only a very superficial overview that is intended to bring the reader up to speed on the basics. See Briggs (2014) and Weirich (2016) and references therein for more detailed introductions.

Evidential decision theory (Ahmed, 2014; Jeffrey, 1990), considers the probability of the percept e *conditional on* taking the action a:

$$\arg\max_{a\in\mathcal{A}} \sum_{e\in\mathcal{E}} \xi(e \mid a)\, u(e) \quad \text{with} \quad \xi(e \mid a) = \sum_{s\in\mathcal{S}} \xi(e \mid s,a)\xi(s \mid a) \tag{EDT}$$

Causal decision theory has several formulations (Gibbard and Harper, 1978; Joyce, 1999; Lewis and Papadimitriou, 1981; Skyrms, 1982); we use the one given in (Skyrms, 1982), with Pearl's calculus of causality (Pearl, 2009). According to CDT, the probability of a percept e is given by the *causal intervention* of performing action a on the causal graph from Figure 10.2:

$$\arg\max_{a\in\mathcal{A}} \sum_{e\in\mathcal{E}} \xi(e \mid \texttt{do}(a))\, u(e) \quad \text{with} \quad \xi(e \mid \texttt{do}(a)) = \sum_{s\in\mathcal{S}} \xi(e \mid s,a)\xi(s) \tag{CDT}$$

where $\xi(e \mid \texttt{do}(a))$ follows from definition of the do-operator (Section 4.1 on Page 46) and marginalization of s.

The difference between CDT and EDT is how the action affects the belief about the hidden state. EDT assigns credence $\xi(s \mid a)$ to the hidden state s if action a is taken, while CDT assigns credence $\xi(s)$. A common argument for CDT is that an action under

my direct control should not influence my belief about things that are not causally affected by the action. Hence $\xi(s)$ should be my belief in s, and not $\xi(s \mid a)$. (By assumption, the action does not *causally* affect the hidden state.) EDT might reply that if action a does not have the same likelihood under all hidden states s, then action a should indeed inform me about the hidden state, regardless of causal connection. The following two classical examples from the decision theory literature describe situations where CDT and EDT disagree. A formal definition of these examples can be found in Appendix 10.A.

Example 10.1 (Newcomb's Problem (Nozick, 1969))**.** In Newcomb's Problem there are two boxes: an opaque box that is either empty or contains one million dollars and a transparent box that contains one thousand dollars. The agent can choose between taking only the opaque box ('one-boxing') and taking both boxes ('two-boxing'). The content of the opaque box is determined by a prediction about the agent's action by a very reliable predictor: if the agent is predicted to one-box, the box contains the million, and if the agent is predicted to two-box, the box is empty. In Newcomb's problem EDT prescribes one-boxing because one-boxing is evidence that the box contains a million dollars. In contrast, CDT prescribes two-boxing because two-boxing dominates one-boxing: in either case we are a thousand dollars richer, and our decision cannot causally affect the prediction. Newcomb's problem has been raised as a critique to CDT, but many philosophers insist that two-boxing is in fact the rational choice,[4] even if it means you end up poor.

Note how the decision depends on whether the action influences the belief about the hidden state (the contents of the opaque box) or not. ♢

Newcomb's problem may appear as an unrealistic thought experiment. However, we argue that problems with similar structure are fairly common. The main structural requirement is that $\xi(s \mid a) \neq \xi(s)$ for some state or event s that is not causally affected by a. In Newcomb's problem the predictor's ability to guess the action induces an 'information link' between actions and hidden states. If the stakes are high enough, the predictor does not have to be much better than random in order to generate a *Newcomblike decision problem.* Consider for example spouses predicting the faithfulness of their partners, employers predicting the trustworthiness of their employees, or parents predicting their children's intentions. For AIs, the potential for accurate predictions is even greater, as the predictor may have access to the AI's source code. Although rarely perfect, all of these predictions are often substantially better than random.

[4] In a 2009 survey, 31.4% of philosophers favored two-boxing, and 21.3% favored one-boxing (931 responses); see `http://philpapers.org/surveys/results.pl`. Is that the reason there are so few wealthy philosophers?

To counteract the impression that EDT is generally superior to CDT, we also discuss the *toxoplasmosis problem.*

Example 10.2 (Toxoplasmosis Problem[5] (Altair, 2013))**.** This problem takes place in a world in which there is a certain parasite that causes its hosts to be attracted to cats, in addition to uncomfortable side effects. The agent is handed an adorable little kitten and is faced with the decision of whether or not to pet it. Petting the kitten feels nice and therefore yields more utility than not petting it. However, people suffering from the parasite are more likely to pet the kitten. Petting the kitten is evidence of having the parasite, so EDT recommends against it. CDT correctly observes that petting the kitten does not *cause* the parasite, and is therefore in favor of petting. ♢

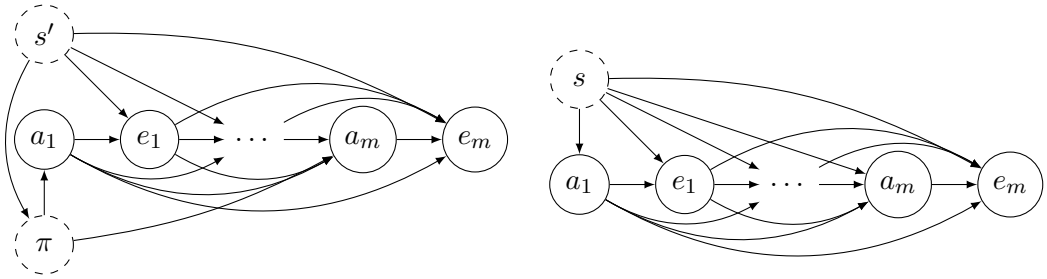
Newcomb's problem (Example 10.1) and the toxoplasmosis problem (Example 10.2) cannot be properly formalized in SDT, because SDT requires the percept-probabilities ξ_a to be specified, but it is not clear what the right choice of ξ_a would be. However, both CDT and EDT can be recast in the context of (SDT) by setting ξ_a to be $\xi(\cdot \mid \texttt{do}(a))$ and $\xi(\cdot \mid a)$ respectively. Thus we could say that the formulation given by Savage needs a specification of the environment that tells us whether to act evidentially, causally, or otherwise.

10.3. The Sequential Physicalistic Model

In this section we formally specify the physicalistic model depicted in Figure 10.1, and briefly discuss problems with time consistency. Returning to the sequential setup of previous chapters, the agent will choose actions $a_t \in \mathcal{A}$ and receive a percepts $e_t \in \mathcal{E}$ for multiple time steps. Rather than considering infinite interactions as in previous chapters, we will restrict ourselves to finite interactions of m time steps. This is to avoid some technicalities. After m time steps, the agent gets utility $\tilde{u}(æ_{1:m})$ out of the interaction. A *history* is an element of $(\mathcal{A} \times \mathcal{E})^*$. We use $æ \in \mathcal{A} \times \mathcal{E}$ to denote one interaction cycle, and $æ_{<t}$ to denote a history of length $t-1$. The percepts between time t and time m are denoted $e_{t:m}$. A *policy* is a function that maps a history $æ_{<t}$ to the next action a_t. For simplicity, we will restrict ourselves to deterministic policies in this chapter.

A causal graph of the agent's model is given in Figure 10.3a. To make notation and derivations simpler, we will aggregate the agent's policy and the hidden state into one

[5] Historically, this problem has been known as the *smoking lesion problem* (Egan, 2007). We consider the smoking lesion formulation confusing, because today it is universally known that smoking *does* cause lung cancer.



(a) Explicit representation of the agent's uncertainty about itself.

(b) Aggregating all unobserved variables into the hidden state $s = (s', \pi)$.

Figure 10.3.: Two representations of the causal graph for the agent's model ξ of a sequential environment. Each action a_t and each percept e_t is represented by a node in the causal graph. Actions and percepts affect all subsequent actions and percepts: causality follows time. The hidden state s is only ever indirectly (partially) observed, and the agent only has partial knowledge about its own configuration π.

node s, as shown in Figure 10.3b. The aggregated model ξ causally factorizes as:

$$\xi(s, æ_{<t}) = \xi(s) \prod_{k=1}^{t-1} \xi(a_k \mid s, æ_{<k}) \xi(e_k \mid s, æ_{<k} a_k) \tag{10.1}$$

for any $1 \leq t \leq m$. The distribution $\xi(a_t \mid s, æ_{<t})$ gives the likelihood of the agent's own actions provided a hidden state $s \in \mathcal{S}$ (for example, the prior probability of an infected agent petting the kitten in the toxoplasmosis problem above). For technical reasons, this distribution must always leave some uncertainty about the actions: if the environment model assigned probability zero for an action a', the agent could not deliberate taking action a' since a' could not be conditioned on. Formally, we require $\xi(\cdot \mid s)$ to be *action-positive* for all $s \in \mathcal{S}$:

$$\forall æ_{<t} a_t \in (\mathcal{A} \times \mathcal{E})^* \times \mathcal{A}\colon \quad \xi(æ_{<t} \mid s) > 0 \implies \xi(a_t \mid s, æ_{<t}) > 0. \tag{10.2}$$

The distribution ξ is a *model* of the environment, a belief held by the agent, but not the distribution from which the actual history is drawn. The actual history is distributed according to the true environment distribution. Because the environment contains the agent, the agent's algorithm might get modified by it and the actions that the agent actually ends up taking might not be the actions that were planned. In the end, model and reality will disagree: for example, we simultaneously assume the agent's policy π to be deterministic and the environment model to be action positive. Nevertheless, we assume the given environment model ξ is *accurate* in the sense that it faithfully represents

the environment in the ways relevant to the agent. In other words, we are interested in problems that arise during planning, not problems that arise due to poor modeling.

10.4. Sequential Decision Theory

In this section, we extend CDT and EDT to the sequential setting defined in Section 10.3. Evidential extensions are discussed first (Section 10.4.1), thereafter causal extensions (Section 10.4.2). In Section 10.4.3 we more closely analyze the differences. In Chapters 5 to 8 we always surrounded the agent's policy with a `do`-operator, which means that the distribution over the agent's policies $\xi(\pi_t)$ was never utilized. This section will investigate the consequences of letting the agent use its belief about its own policy to make inferences about the environment.

10.4.1. Sequential Evidential Decision Theory

In one-shot settings, evidential decision theory assigns probability $\xi(e \mid a)$ to action a resulting in percept e (Section 10.2.2). There are two ways to generalize this to the sequential setting, depending on whether we use only the next action or the whole future policy as evidence for the next percept.

Definition 10.3 (Action-Evidential Decision Theory)**.** The *action-evidential value of a policy π with lifetime m in environment ξ given history $æ_{<t}a_t$* is

$$V_{\xi,m}^{\text{aev},\pi}(æ_{<t}a_t) := \sum_{e_t} \xi(e_t \mid æ_{<t}a_t)\Big(u(e_t) + V_{\xi,m}^{\text{aev},\pi}(æ_{<t}a_te_t)\Big) \tag{SAEDT}$$

and $V_{\xi,m}^{\text{aev},\pi}(æ_{<t}a_t) := 0$ for $t > m$. *Sequential Action-Evidential Decision Theory (SAEDT)* prescribes adopting an optimal and time consistent policy π for $V_{\xi,m}^{\text{aev}}$.

It may be argued that SAEDT does not take all available (deliberative) information into account. When considering the consequences of an action, future developments of the environment-policy interactions could also be used as evidence. That is, we could condition not only on the next action, but on the future policy as a whole (within the lifetime). In order to define conditional probabilities with respect to (deterministic) policies, we define the following events. For a given policy π, let $\Pi_{t:m}$ be the set of all strings consistent with π between time step t and m:

$$\Pi_{t:m} := \{æ_{1:\infty} \mid \forall t \leq i \leq m\colon \pi(æ_{<i}) = a_i\}$$

The likelihood of a next percept e_t provided a history $æ_{<t}$ and a (future) policy π followed from time step t until lifetime m (denoted $\pi_{t:m}$) is then defined as

$$\xi(e_t \mid æ_{<t}, \pi_{t:m}) := \xi(e_t \mid æ_{<t} \cap \Pi_{t:m}). \tag{10.3}$$

The notation $\pi_{t:m}$ has a slightly different interpretation here than in Chapters 5 to 8. In previous chapters, $\pi_{t:m}$ was a random variable due to the possibility of policy self-corruption. Here instead the uncertainty comes from the agent not fully knowing its own policy.

Equation (10.3) is an *atemporal* conditional because we are conditioning on future actions up until the end of the agent's lifetime. The conditional (10.3) is well-defined because we only take the actions from time step t to m into account; conditioning on policies with infinite lifetime leads to technical problems because such policies may have ξ-measure zero.

We are now ready to define the policy-evidential extension of evidential decision theory.

Definition 10.4 (Policy-Evidential Decision Theory)**.** The *policy-evidential value of a policy π with lifetime m in environment ξ given history $æ_{<t}a_t$* is

$$V^{\text{pev},\pi}_{\xi,m}(æ_{<t}a_t) := \sum_{e_t} \xi(e_t \mid æ_{<t}a_t, \pi_{t+1:m}) \cdot \Big(u(e_t) + V^{\text{pev},\pi}_{\xi,m}(æ_{<t}a_te_t)\Big) \tag{SPEDT}$$

and $V^{\text{pev},\pi}_{\xi,m}(æ_{<t}) := 0$ for $t > m$. *Sequential Policy-Evidential Decision Theory (SPEDT)* prescribes adopting an optimal and time consistent policy π for $V^{\text{pev}}_{\xi,m}$.

For one-step decisions ($m = t + 1$), SAEDT and SPEDT coincide.

Comparing action- and policy-evidential decision theory. To all our embedded agents, past actions constitute evidence about the hidden state. For evidential agents, this principle is extended to future actions. SAEDT and SPEDT differ in how far they extend it. The action-evidential agent only updates its belief on the action about to take place. In that sense, it only updates its belief about the next percept on events taking place *before* this percept. The policy-evidential agent takes the principle much further, using "thought-experiments" of what action it *would take in hypothetical situations*, most of which will never be realized. This is illustrated in the next example.

Example 10.5 (Sequential Toxoplasmosis)**.** In our sequential variation of the toxoplasmosis problem (Example 10.2) the agent has some probability of encountering a kitten. Additionally, the agent has the option of seeing a doctor (for a fee) and getting tested for the parasite, which can then be safely removed. In the very beginning, an SPEDT

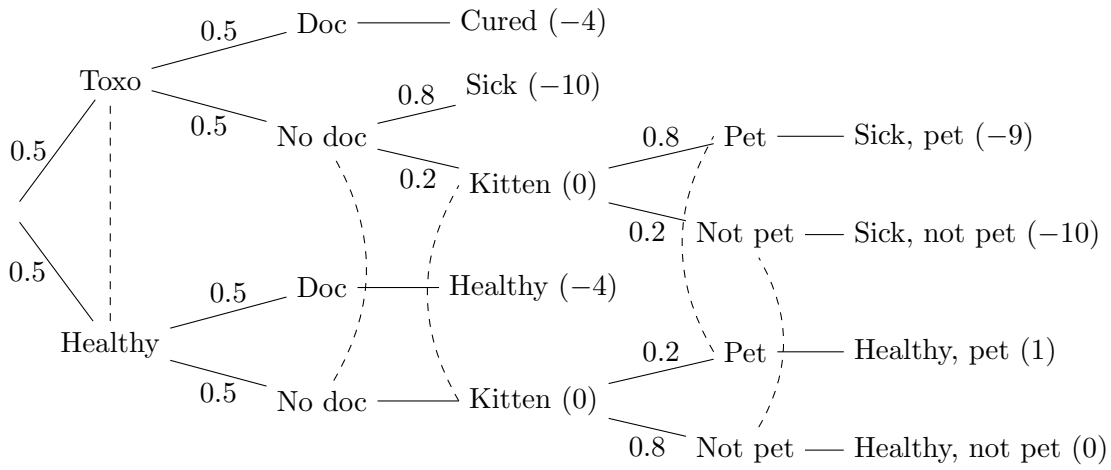


Figure 10.4.: One formalization of the sequential toxoplasmosis problem. Dashed lines connect states indistinguishable to the agent. The numbers on the edges indicate probabilities of the environment model ξ, and the numbers in parenthesis indicate utilities of the associated percepts. In the first step, the environment selects the hidden state that is unknown to the agent. The agent then decides whether to go to the doctor. If he does not go, he may encounter a kitten which he can choose to pet or not. SAEDT and SPEDT will disagree whether going to the doctor is the best option in this scenario. Appendix 10.A contains the full calculations.

agent updates his belief on the fact that if he encountered a kitten, he would not pet it, which lowers the probability that he has the parasite and makes seeing the doctor unattractive. An SAEDT agent only updates his belief about the parasite when he actually encounters a kitten, and thus prefers seeing the doctor. See Figure 10.4 for more details and a graphical illustration. ♢

The observant reader may ask whether SPEDT could be enticed to make some percepts unlikely by choosing improbable actions subsequent to them. For example, could an SPEDT agent decide on a policy of selecting highly improbable actions in case it rained to make histories with rain less likely? The answer is no, as most such policies would not be time consistent. If it does rain, the highly improbable action would usually not the best one, and so the policy would not be prescribed by Definition 10.4.

10.4.2. Sequential Causal Decision Theory

In sequential causal decision theory we ask what would happen if we causally intervened on the node a_t of the next action and fix it to $\pi(æ_{<t})$ according to the policy π. This is expressed by the notation $\texttt{do}(a_t := \pi(æ_{<t}))$, or $\texttt{do}(\pi(æ_{<t}))$ for short.

Definition 10.6 (Sequential Causal Decision Theory)**.** The *causal value of a policy* π *with lifetime* m *in environment* ξ *given history* $æ_{<t}a_t$ is

$$V^{\text{cau},\pi}_{\xi,m}(æ_{<t}a_t) := \sum_{e_t\in\mathcal{E}} \xi(e_t \mid æ_{<t}, \texttt{do}(a_t))\Big(u(e_t) + V^{\text{cau},\pi}_{\xi,m}(æ_{<t}a_te_t)\Big) \tag{SCDT}$$

and $V^{\text{cau},\pi}_{\xi,m}(æ_{<t}a_t) := 0$ for $t > m$. *Sequential Causal Decision Theory (SCDT)* prescribes adopting an optimal and time consistent policy π for $V^{\text{cau}}_{\xi,m}$.

For sequential evidential decision theory we discussed two versions (SAEDT) and (SPEDT), based on next action and future policy respectively. In (SCDT) we perform the causal intervention $\texttt{do}(a_t := \pi(æ_{<t}))$. We could also consider a policy-causal decision theory by replacing $\xi(e_t \mid æ_{<t}, \texttt{do}(a_t))$ with $\xi(e_t \mid æ_{<t}, \texttt{do}(\pi_{t:m}))$ in Definition 10.6. The causal intervention $\texttt{do}(\pi_{t:m}))$ of a policy π between time step t and time step m is defined as as

$$\xi(e_t \mid æ_{<t}, \texttt{do}(\pi_{t:m})) := \sum_{e_{t+1:m}} \xi(e_{t:m} \mid æ_{<t}, \texttt{do}(a_t := \pi(æ_{<t}), \ldots, a_m := \pi(æ_{<m}))). \tag{10.4}$$

This most closely resembles the decision principle we employed in Chapters 5 to 8. However, since the interventions are causal, we do not get any extra evidence from the future interventions. Therefore policy-causal decision theory is the same as action-causal decision theory:

Proposition 10.7 (Policy-Causal = Action-Causal)**.** *For all histories* $æ_{<t} \in (\mathcal{A}\times\mathcal{E})^*$ *and all* $e_t \in \mathcal{E}$*, we have* $\xi(e_t \mid æ_{<t}, \texttt{do}(\pi_{t:m})) = \xi(e_t \mid æ_{<t}, \texttt{do}(\pi(æ_{<t})))$.

We defer the proof to the end of this section. The following two examples illustrate the difference between SCDT and SAEDT/SPEDT in sequential settings.

Example 10.8 (Newcomb with Precommitment)**.** In this variation of Newcomb's problem (Example 10.1) the agent first has the option to pay \$300,000 to sign a contract that binds the agent to pay \$2000 in case of two-boxing. An SAEDT or SPEDT agent knows that he will one-box anyways and hence has no need for the contract. An SCDT agent knows that she favors two-boxing, but signs the contract only if this occurs before the prediction is made (so it has a chance of causally affecting the prediction). With the contract in place, one-boxing is the dominant action, and thus the SCDT agent is predicted to one-box. ♢

Example 10.9 (Newcomb with Looking)**.** In this variation of Newcomb's problem (Example 10.1) the agent may look into the opaque box before making the decision which

box to take. An SCDT agent is indifferent towards looking because she will take both boxes anyways. However, an SAEDT or SPEDT agent will avoid looking into the box, because once the content is revealed he two-boxes. ♢

10.4.3. Expansion over the Hidden State

The difference between sequential versions of EDT and CDT is how they update their prediction of a next percept e_t (Definitions 10.3, 10.4 and 10.6). The following proposition expands the different beliefs in terms of the hidden state.

Proposition 10.10. *For all histories $æ_{<t}a_te_t \in (\mathcal{A} \times \mathcal{E})^*$ the following holds for the next-percept beliefs of SAEDT, SPEDT and SCDT respectively:*

$$\xi(e_t \mid æ_{<t}a_t) = \sum_{s \in \mathcal{S}} \xi(s \mid æ_{<t}a_t)\xi(e_t \mid s, æ_{<t}a_t) \tag{10.5}$$

$$\xi(e_t \mid æ_{<t}, \pi_{t:m}) = \sum_{s \in \mathcal{S}} \xi(s \mid æ_{<t}, \pi_{t:m})\xi(e_t \mid s, æ_{<t}, \pi_{t:m}) \tag{10.6}$$

$$\xi(e_t \mid æ_{<t}, \texttt{do}(a_t)) = \sum_{s \in \mathcal{S}} \xi(s \mid æ_{<t})\xi(e_t \mid s, æ_{<t}a_t) \tag{10.7}$$

Proof. For the action-evidential conditional we take the joint distribution with s, and then split off e_t:

$$\begin{aligned}\xi(e_t \mid æ_{<t}a_t) = \frac{\sum_{s \in \mathcal{S}} \xi(s, æ_{<t}a_te_t)}{\xi(æ_{<t}a_t)} &= \frac{\sum_{s \in \mathcal{S}} \xi(s, æ_{<t}a_t)\xi(e_t \mid s, æ_{<t}a_t)}{\xi(æ_{<t}a_t)} \\ &= \sum_{s \in \mathcal{S}} \xi(s \mid æ_{<t}a_t)\xi(e_t \mid s, æ_{<t}a_t)\end{aligned}$$

Similarly for the policy-evidential conditional:

$$\begin{aligned}\xi(e_t \mid æ_{<t}, \pi_{t:m}) &= \frac{\sum_{s \in \mathcal{S}} \xi(s, æ_{<t}\pi(æ_{<t})e_t, \pi_{t+1:m})}{\xi(æ_{<t}, \pi_{t:m})} \\ &= \frac{\sum_{s \in \mathcal{S}} \xi(s, æ_{<t}\pi(æ_{<t}), \pi_{t+1:m})\xi(e_t \mid s, æ_{<t}\pi(æ_{<t}), \pi_{t+1:m})}{\xi(æ_{<t}, \pi_{t:m})} \\ &= \frac{\sum_{s \in \mathcal{S}} \xi(s, æ_{<t}, \pi_{t:m})\xi(e_t \mid s, æ_{<t}\pi(æ_{<t}), \pi_{t+1:m})}{\xi(æ_{<t}, \pi_{t:m})} \\ &= \sum_{s \in \mathcal{S}} \xi(s \mid æ_{<t}, \pi_{t:m})\xi(e_t \mid s, æ_{<t}\pi(æ_{<t}), \pi_{t+1:m}) \\ &= \sum_{s \in \mathcal{S}} \xi(s \mid æ_{<t}, \pi_{t:m})\xi(e_t \mid s, æ_{<t}, \pi_{t:m})\end{aligned}$$

For the causal conditional we turn to the rules of the do-operator (Pearl, 2009, Thm.

3.4.1). The first equality below holds by definition. In the denominator of the second equality we can use Rule 3 (deletion of actions) to remove $\texttt{do}(a_t)$ because the do-operator removes all incoming edges to a_t and makes a_t independent of the history $æ_{<t}$. In the numerator of the second equality we use the definition of do (4.3):

$$\begin{aligned}\xi(e_t \mid æ_{<t}, \texttt{do}(a_t)) &= \frac{\xi(æ_{<t}, e_t \mid \texttt{do}(a_t))}{\xi(æ_{<t} \mid \texttt{do}(a_t))} \\ &= \frac{\sum_{s\in\mathcal{S}} \xi(s, æ_{<t})\xi(e_t \mid s, æ_{<t}a_t)}{\xi(æ_{<t})} \\ &= \sum_{s\in\mathcal{S}} \xi(s \mid æ_{<t})\xi(e_t \mid s, æ_{<t}a_t) \qquad \square\end{aligned}$$

Proposition 10.10 shows that between SCDT and SAEDT, the difference in opinion about e_t only depends on differences in their (acausal) *posterior belief* $\xi(s \mid \ldots)$ about the hidden state. SCDT and SAEDT thus become equivalent in scenarios where there is only one hidden state s^* with $\xi(s^*) = 1$, as this renders $\xi(s^* \mid æ_{<t}) = \xi(s^* \mid æ_{<t}a_t) = \xi(s^*) = 1$. SPEDT, on the other hand, may disagree with the other two also after a hidden state has been fixed.

From a problem modeler's perspective, it is also instructive to consider the effect of moving uncertainty between the hidden state and environmental stochasticity. For two different environment models ξ and ξ', the action and percept probabilities may be identical (i.e., $\xi(a_t \mid æ_{<t}) = \xi'(a_t \mid æ_{<t})$ and $\xi(e_t \mid æ_{<t}a_t) = \xi'(e_t \mid æ_{<t}a_t)$) even though ξ and ξ' have non-isomorphic sets of hidden states $\mathcal{S}$ and $\mathcal{S}'$. For example, given any ξ, an environment model ξ' with a single hidden state s_0, $\xi'(s_0) = 1$, may be constructed from ξ by $\xi'(s_0, æ_{<t}) := \sum_{s\in\mathcal{S}} \xi(s, æ_{<t})$. The transformation will not affect SAEDT and SPEDT, as the definitions of their value functions only depends on the 'observable' action- and percept-probabilities $\xi(a_t \mid æ_{<t})$ and $\xi(e_t \mid æ_{<t}a_t)$ which are preserved between ξ and ξ'. But the transformation will change SCDT's behavior in any ξ where SCDT disagrees with SAEDT, as SCDT and SAEDT are equivalent in ξ' that only has a single hidden state. That SCDT depends on what uncertainty is captured by the hidden state is unsurprising given that the hidden state has a special place in the causal structure of the problem. Ultimately, the modeler must decide what uncertainty to put in the hidden state, and what to attribute to environmental stochasticity. A general principle for how to do this is still an open question (Soares and Fallenstein, 2015b).

The value functions of SAEDT, SPEDT and SCDT can be rewritten in the following *iterative forms*, where the latter form uses Proposition 10.10. Numbers above equality

signs reference a justifying equation. Let $a_i := \pi(æ_{<i})$ for $i \geq t$:

$$V_{\xi,m}^{\text{aev},\pi}(æ_{<t}) = \sum_{k=t}^{m} \sum_{e_{t:k}} u(e_k) \prod_{i=t}^{k} \xi(e_i \mid æ_{<i} a_i) \tag{10.8}$$

$$\overset{(10.5)}{=} \sum_{k=t}^{m} \sum_{e_{t:k}} u(e_k) \prod_{i=t}^{k} \sum_{s \in \mathcal{S}} \xi(s \mid æ_{<i} a_i) \xi(e_i \mid s, æ_{<i} a_i) \tag{10.9}$$

$$V_{\xi,m}^{\text{pev},\pi}(æ_{<t}) = \sum_{k=t}^{m} \sum_{e_{t:k}} u(e_k) \prod_{i=t}^{k} \xi(e_i \mid æ_{<i}, \pi_{i:m}) \tag{10.10}$$

$$\overset{(10.6)}{=} \sum_{k=t}^{m} \sum_{e_{t:k}} u(e_k) \prod_{i=t}^{k} \sum_{s \in \mathcal{S}} \xi(s \mid æ_{<i} \pi_{i:m}) \xi(e_i \mid s, æ_{<i}, \pi_{i:m}) \tag{10.11}$$

$$V_{\xi,m}^{\text{cau},\pi}(æ_{<t}) = \sum_{k=t}^{m} \sum_{e_{t:k}} u(e_k) \prod_{i=t}^{k} \xi(e_i \mid æ_{<i}, \texttt{do}(a_i)) \tag{10.12}$$

$$\overset{(10.7)}{=} \sum_{k=t}^{m} \sum_{e_{t:k}} u(e_i) \prod_{i=t}^{k} \sum_{s \in \mathcal{S}} \xi(s \mid æ_{<i}) \xi(e_i \mid s, æ_{<i} a_i) \tag{10.13}$$

Proof of Proposition 10.7. By the definition (10.4) of $\texttt{do}(\pi_{t:m})$,

$$\begin{aligned}
\xi(e_t \mid æ_{<t}, \texttt{do}(\pi_{t:m})) &= \sum_{e_{t+1:m}} \xi(e_{t:m} \mid æ_{<t}, \texttt{do}(a_t := \pi(æ_{<t}), \ldots, a_m := \pi(æ_{<m}))) \\
&= \sum_{s, e_{t+1:m}} \xi(s \mid æ_{<t}) \xi(e_{t:m} \mid s, æ_{<t}, \texttt{do}(\pi(æ_{<t}), \ldots, \pi(æ_{<m}))) \\
&\overset{(4.3)}{=} \sum_{s, e_{t+1:m}} \xi(s \mid æ_{<t}) \prod_{i=t}^{m} \xi(e_i \mid s, æ_{<i} \pi(æ_{<i})) \\
&= \sum_{s} \xi(s \mid æ_{<t}) \xi(e_t \mid s, æ_{<t} \pi(æ_{<t})) \\
&\overset{(10.7)}{=} \xi(e_t \mid æ_{<t}, \texttt{do}(\pi(æ_{<t})))
\end{aligned}$$

The second equality follows from the equivalence $P(\,\cdot\,) = \sum_s P(s) P(\,\cdot \mid s)$ applied to the distribution $\xi(\,\cdot \mid æ_{<t}, \texttt{do}(a_t := \pi(æ_{<t}), \ldots, a_m := \pi(æ_{<m})))$, and the third equality by (repeated) application of (4.3) to $\xi(æ_{t:m} \mid s, æ_{<t}) = \prod_{i=t}^{m} \xi(a_i \mid s, æ_{<i}) \xi(e_i \mid s, æ_{<i} a_i)$. □

	(SAEDT)	(SPEDT)	(SCDT)
Nwcb	*1-box*	*1-box*	2-box
Nwcb w/ precommit	*not commit, 1-box*	*not commit, 1-box*	commit, 1-box
Nwcb w/ looking	not look, 1-box	not look, 1-box	indifferent, 2-box
Toxoplasmosis	not pet	not pet	*pet*
Seq. Toxoplasmosis	doc, not pet	no doc, not pet	*doc, pet*

Table 10.1.: Decisions made by (SAEDT), (SPEDT), and (SCDT) in Examples 10.1, 10.2, 10.5, 10.8 and 10.9. The latter three examples are sequential. Winning moves are in italics; in Newcomb with looking the winning move is to be indifferent and one-box. Because Savage decision theory is dualistic, these problems cannot be properly formalized in it.

10.5. Discussion

This chapter is a first stab at the problem of how physicalistic agents should make sequential decisions. CDT and EDT provide an existing basis for non-dualistic decision making, which we extended to the sequential setting. There are two natural ways for making sequential evidential decisions: do I update my beliefs about the hidden state based on my next action ('what I do next', (SAEDT)) or my whole policy ('the kind of agent I am', (SPEDT))? By Proposition 10.7, this distinction does not exist for causal decision theory, because with that theory the agent does not consider its own actions evidence at all. Therefore we have only one version of sequential causal decision theory, (SCDT).

To illustrate the differences between the decision theories, we discussed three variants of Newcomb's problem (Examples 10.1, 10.8 and 10.9) and two variants of the toxoplasmosis problem (Examples 10.2 and 10.5). The formal specification of these examples can be found in Appendix 10.A. We have also implemented SCDT, SAEDT, and SPEDT; Table 10.1 shows their behavior on the mentioned examples.[6]

So which decision theory is better? The answer to this question depends on which decision you consider to be *correct* (or even *rational*) in each of the problems. We posit that ultimately, what counts is not whether your decision algorithm is theoretically pleasing, but *whether you win*. Winning means getting the most utility. If maximizing utility involves making crazy decisions, then this is what you should do!

In Newcomb's problem, winning means one-boxing, because you end up richer. In the toxoplasmosis problem, winning means petting the kitten, because that yields more utility. (S)CDT performs suboptimally in the Newcomb variations, while the evidential

[6]Source code available at `http://jan.leike.name/`.

decision theories perform suboptimally in the toxoplasmosis variations. This entails that neither CDT nor EDT are the final answer to the problem of non-dualistic decision making.

Furthermore, neither CDT nor EDT agents are fully physicalistic: they do not model the environment to contain themselves (Soares and Fallenstein, 2015b). For example, when playing a prisoner's dilemma against your own source code (Lavictoire et al., 2014), your opponent defects if and only if you defect. This *logical* connection between your action and your opponent's is disregarded in the formalization based on causal graphical models that we discuss here because it is not causal.

Functional decision theory (Yudkowsky and Soares, 2017) is a recent attempt to formalize an improvement over CDT and EDT. As with CDT and EDT prior to our work, it has so far only been explored in one-shot settings. Future work may investigate its behavior in sequential settings.

10.A. Examples

This section contains the formal calculations for Examples 10.1, 10.2, 10.5, 10.8 and 10.9. These calculations are also available as Python code at `http://jan.leike.name/`.

Example 10.11 (Newcomb's Problem)**.** This is a formalization of Example 10.1.

- $\mathcal{S} := \{E, F\}$ where E means the opaque box is empty and F means the opaque box is full
- $\mathcal{A} := \{B_1, B_2\}$ where B_1 means one-boxing and B_2 means two-boxing
- $\mathcal{E} := \{O_0, O_T, O_M, O_{MT}\}$
- $u(O_0) := 0$, $u(O_T) := 1{,}000$, $u(O_M) := 1{,}000{,}000$, $u(O_{MT}) := 1{,}001{,}000$

Let $\varepsilon > 0$ be a small constant denoting the accuracy of the predictor. Because the environment has to assign non-zero probability to all actions, ε must be strictly positive. The environment's distribution ξ is defined as follows.

$$\begin{aligned} \xi(E) = \xi(F) &= 0.5 & \xi(O_T \mid E, B_2) &= 1 \\ \xi(B_1 \mid F) = \xi(B_2 \mid E) &= 1-\varepsilon & \xi(O_0 \mid E, B_1) &= 1 \\ \xi(B_1 \mid E) = \xi(B_2 \mid F) &= \varepsilon & \xi(O_{MT} \mid F, B_2) &= 1 \\ & & \xi(O_M \mid F, B_1) &= 1 \end{aligned}$$

By Bayes' rule,

$$\xi(F \mid B_1) = \frac{\xi(B_1 \mid F)\xi(F)}{\sum_{s\in\mathcal{S}} \xi(B_1 \mid s)\xi(s)} = \frac{\frac{1}{2}(1-\varepsilon)}{\frac{1}{2}(1-\varepsilon) + \frac{1}{2}\varepsilon} = (1-\varepsilon)$$

which also gives $\xi(E \mid B_1) = \varepsilon$. Similarly, $\xi(F \mid B_2) = \varepsilon$ and $\xi(E \mid B_2) = 1-\varepsilon$.

For EDT we use equation (EDT) to compute the value of an action. Since the percept e_1 is generated deterministically, $\xi(e \mid s, a)$ only attains values 0 or 1. We therefore omit it in the calculation below. For action B_1 we get

$$\begin{aligned} V_{\xi,1}^{\text{evi},B_1} := \sum_{e\in\mathcal{E}} \xi(e \mid B_1)u(e) &= \sum_{e\in\mathcal{E}}\sum_{s\in\mathcal{S}} \xi(e \mid s, B_1)\xi(s \mid B_1)u(e) \\ &= \xi(E \mid B_1)u(O_0) + \xi(F \mid B_1)u(O_M) \\ &= \varepsilon \cdot 0 + (1-\varepsilon)\cdot 1{,}000{,}000 \end{aligned}$$

For action B_2 we get

$$\begin{aligned} V_{\xi,1}^{\mathrm{evi},B_2} := \sum_{e\in\mathcal{E}} \xi(e \mid B_2)u(e) &= \sum_{e\in\mathcal{E}}\sum_{s\in\mathcal{S}} \xi(e \mid s, B_2)\xi(s \mid B_2)u(e) \\ &= \xi(E \mid B_2)u(O_T) + \xi(F \mid B_2)u(O_{MT}) \\ &= (1-\varepsilon)\cdot 1,000 + \varepsilon \cdot 1,001,000 \\ &= 1,000 + \varepsilon \cdot 1,000,000 \end{aligned}$$

For $\varepsilon < 0.4995$ (just slightly better than random guessing), we get that EDT favors B_1 over B_2:

$$V_{\xi,1}^{\mathrm{evi},B_1} = (1-\varepsilon)\cdot 1,000,000 > 500,500 > 1,000 + \varepsilon \cdot 1,000,000 = V_{\xi,1}^{\mathrm{evi},B_2}$$

For CDT we use equation (CDT) to compute the value of an action. For action B_1 we get

$$\begin{aligned} V_{\xi,1}^{\mathrm{cau},B_1} := \sum_{e\in\mathcal{E}} \xi(e \mid \mathtt{do}(B_1))u(e) &= \sum_{e\in\mathcal{E}}\sum_{s\in\mathcal{S}} \xi(e \mid s, B_1)\xi(s)u(e) \\ &= \xi(E)u(O_0) + \xi(F)u(O_M) \\ &= 0.5\cdot 0 + 0.5 \cdot 1,000,000 = 500,000 \end{aligned}$$

For action B_2 we get

$$\begin{aligned} V_{\xi,1}^{\mathrm{cau},B_2} := \sum_{e\in\mathcal{E}} \xi(e \mid \mathtt{do}(B_2))u(e) &= \sum_{e\in\mathcal{E}}\sum_{s\in\mathcal{S}} \xi(e \mid s, B_2)\xi(s)u(e) \\ &= \xi(E)u(O_T) + \xi(F)u(O_{MT}) \\ &= 0.5\cdot 1,000 + 0.5 \cdot 1,001,000 = 500,500 \end{aligned}$$

We get that CDT favors B_2 over B_1 regardless of the prediction accuracy ε:

$$V_{\xi,1}^{\mathrm{cau},B_1} = 500,000 < 500,500 = V_{\xi,1}^{\mathrm{cau},B_2}$$

Moreover, CDT prefers B_2 *regardless of the prior over* $\xi(E)$. Two-boxing is the dominant action because it yields \$1,000 more regardless of the hidden state. ♢

Example 10.12 (Newcomb with Looking)**.** This is a formalization of Example 10.9; it extends Example 10.11.

In the first time step, the agent gets to choose between looking into the box (L) and not looking (N). If the agent looks, the subsequent percept will be E or F, depending on whether the box is empty (E) or full (F). If the agent does not look, the subsequent percept will be 0. All three of these percepts E, F, and 0 have zero utility.

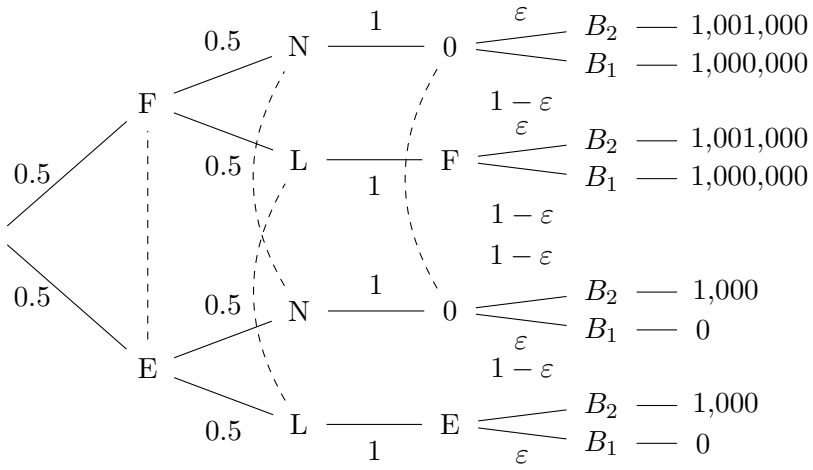
In the second time step the agent chooses to one-box (B_1) or to two-box (B_2). The payoffs are then based on the boxes' contents as in Example 10.11.

- $\mathcal{S} := \{E, F\}$ where E means the opaque box is empty and F means the opaque box is full
- $\mathcal{A} := \{B_1, B_2\}$ where B_1 means one-boxing and B_2 means two-boxing, $L := B_1$ means looking into the box and $N := B_2$ means not looking (the set of actions has to be the same for all time steps)
- $\mathcal{E} := \{E, F, 0, O_0, O_T, O_M, O_{MT}\}$
- $u(O_0) := 0$, $u(O_T) := 1{,}000$, $u(O_M) := 1{,}000{,}000$, $u(O_{MT}) := 1{,}001{,}000$, $u(E) := u(F) := u(0) := 0$

Let $\varepsilon > 0$ be a small constant denoting the prediction accuracy. Because the environment has to assign non-zero probability to all actions, ε must be strictly positive. The environment's distribution ξ is defined as follows. Question marks stand for single actions or percepts whose value is irrelevant.

$$\begin{aligned}
\xi(E) = \xi(F) &= 0.5 & \xi(E \mid E, L) &= 1 \\
\xi(L \mid F) = \xi(L \mid E) &= 0.5 & \xi(0 \mid E, N) &= 1 \\
\xi(N \mid F) = \xi(N \mid E) &= 0.5 & \xi(F \mid F, L) &= 1 \\
\xi(B_1 \mid E, ??) &= \varepsilon & \xi(0 \mid F, N) &= 1 \\
\xi(B_1 \mid F, ??) &= 1 - \varepsilon & \xi(O_0 \mid E, ??B_1) &= 1 \\
\xi(B_2 \mid E, ??) &= 1 - \varepsilon & \xi(O_T \mid E, ??B_2) &= 1 \\
\xi(B_2 \mid F, ??) &= \varepsilon & \xi(O_M \mid F, ??B_1) &= 1 \\
& & \xi(O_{MT} \mid F, ??B_2) &= 1
\end{aligned}$$

The environment's game tree is given as follows, where dashed lines connect states indistinguishable by the agent (also known as *information sets*):



Using Bayes' rule, we calculate the following conditional probabilities of the hidden state given a history a_1 or $a_1e_1a_2$:

$$\begin{aligned}
0.5 &= \xi(E \mid L) = \xi(F \mid L) = \xi(E \mid N) = \xi(F \mid N) \\
1 &= \xi(E \mid LEB_1) = \xi(E \mid LEB_2) = \xi(F \mid LFB_1) = \xi(F \mid LFB_1) \\
\varepsilon &= \xi(E \mid N0B_1) = \xi(F \mid N0B_2) \\
1-\varepsilon &= \xi(E \mid N0B_2) = \xi(F \mid N0B_1)
\end{aligned}$$

Next, we write out the formula for SAEDT for a horizon of 2 based on (10.9). The first percept has no utility, which simplifies the equation.

$$V_{\xi,2}^{\text{aev},\pi} = \sum_{e_{1:2}} u(e_2) \left(\sum_{s \in \mathcal{S}} \xi(s \mid a_1) \xi(e_1 \mid s, a_1) \right) \left(\sum_{s \in \mathcal{S}} \xi(s \mid æ_1 a_2) \xi(e_2 \mid s, æ_1 a_2) \right)$$

where $a_1 = \pi(\epsilon)$ and $a_2 = \pi(æ_1)$. The formula for SPEDT for a horizon of 2 based on (10.11) is as follows.

$$V_{\xi,2}^{\text{pev},\pi} = \sum_{e_{1:2}} u(e_2) \frac{\sum_{s \in \mathcal{S}} \xi(s a_1 e_1 \pi(a_1 e_1))}{\sum_{s \in \mathcal{S}} \sum_{e \in \mathcal{E}} \xi(s a_1 e \pi(a_1 e))} \sum_{s \in \mathcal{S}} \xi(s \mid æ_1 \pi_2) \xi(e_2 \mid s, æ_1 a_2)$$

with $\pi_{1:2}$ and π_2 defined according to (10.3). The formula for SCDT for a horizon of 2 based on (10.13) is as follows.

$$V_{\xi,2}^{\text{cau},\pi} = \sum_{e_{1:2}} u(e_2) \left(\sum_{s \in \mathcal{S}} \xi(s) \xi(e_1 \mid s, a_1) \right) \left(\sum_{s \in \mathcal{S}} \xi(s \mid æ_1) \xi(e_2 \mid s, æ_1 a_2) \right)$$

where $a_1 = \pi(\epsilon)$ and $a_2 = \pi(æ_1)$.

There are six different possible policies:

- Look and always one-box (curious one-boxer)
- Look and always two-box (curious two-boxer)
- Don't look and one-box (incurious one-boxer)
- Don't look and two-box (incurious two-boxer)
- Look and one-box iff the box is empty (paradox-lover)
- Look and one-box iff the box full (fatalistic)

Using the formulas above we can calculate their value. We use $\varepsilon := 0.01$.

	$V_{\xi,2}^{\text{aev},\pi}$	$V_{\xi,2}^{\text{pev},\pi}$	$V_{\xi,2}^{\text{cau},\pi}$
Curious one-boxer	500,000	*990,000*	500,000
Curious two-boxer	501,000	11,000	*501,000*
Incurious one-boxer	*990,000*	*990,000*	500,000
Incurious two-boxer	11,000	11,000	*501,000*
Paradox-lover	500,500	500,500	500,500
Fatalistic	500,500	500,500	500,500

The highest values are displayed in italics. The incurious one-boxer has the highest action-evidential value. The curious one-boxer and the incurious one-boxer have the highest policy-evidential value. However, of these two policies only the incurious one-boxer is a time-consistent policy for SPEDT, because the agent wants to two-box after looking into the box:

$$
\begin{aligned}
V_{\xi,1}^{\text{aev},B_1}(LF) &= V_{\xi,1}^{\text{pev},B_1}(LF) = 1,000,000 \\
V_{\xi,1}^{\text{aev},B_2}(LF) &= V_{\xi,1}^{\text{pev},B_2}(LF) = 1,001,000 \\
V_{\xi,1}^{\text{aev},B_1}(LE) &= V_{\xi,1}^{\text{pev},B_1}(LE) = 0 \\
V_{\xi,1}^{\text{aev},B_2}(LE) &= V_{\xi,1}^{\text{pev},B_2}(LE) = 1,000
\end{aligned}
$$

The curious two-boxer and the incurious two-boxer have the highest causal value, and they are both time-consistent for SCDT. ♢

Example 10.13 (Newcomb with Precommitment)**.** This is a formalization of Example 10.8, it extends Example 10.11.

In the first time step, the agent gets to choose between signing the contract (S) and not signing (N). If the agent signs, the subsequent percept will be C, which costs \$300,000, and the prediction will be updated to one-boxing. If the agent does not sign, the subsequent percept will be 0 with zero utility.

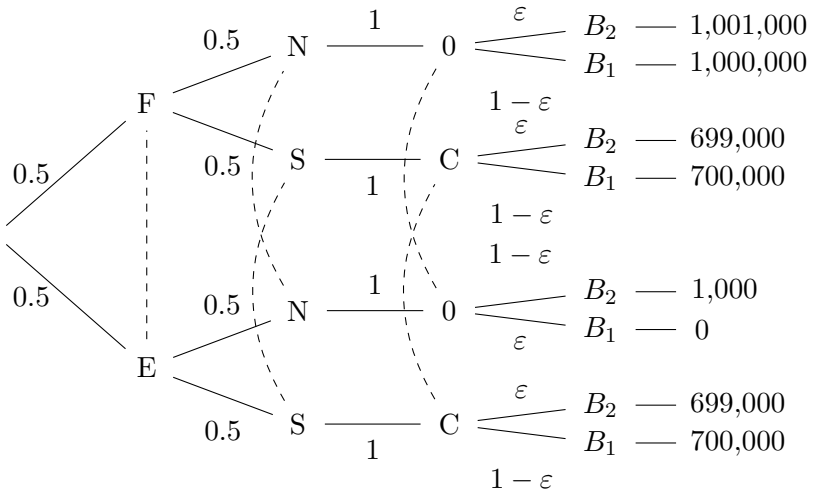
In the second time step the agent chooses to one-box (B_1) or to two-box (B_2). The payoffs are then based on the boxes' contents as in Example 10.11. If the agent signed the contract and chooses two boxes, this incurs an additional cost of \$2,000. Signing the contract overrides the initial prediction and the opaque box is always treated as full.

- $\mathcal{S} := \{E, F\}$ where E means the opaque box is empty and F means the opaque box is full
- $\mathcal{A} := \{B_1, B_2\}$ where B_1 means one-boxing and B_2 means two-boxing, $S := B_1$ means signing the contract and $N := B_2$ means not signing (the set of actions has to be the same for all time steps)
- $\mathcal{E} := \{C, 0, O_0, O_T, O_{-T}, O_M, O_{MT}, O_{M-T}\}$
- $u(O_0) := 0$, $u(O_T) := 1,000$, $u(O_{-T}) := -1,000$ $u(O_M) := 1,000,000$, $u(O_{MT}) := 1,001,000$, $u(O_{M-T}) := 999,000$, $u(C) := -300,000$, $u(0) := 0$

Let $\varepsilon > 0$ be a small constant denoting the prediction accuracy. Because the environment has to assign non-zero probability to all actions, ε must be strictly positive. The environment's distribution ξ is defined as follows. Question marks stand for single actions, percepts, or hidden states whose value is irrelevant.

$$\begin{aligned}
\xi(E) = \xi(F) &= 0.5 & \xi(C \mid E, S) &= 1\\
\xi(S \mid F) = \xi(S \mid E) &= 0.5 & \xi(0 \mid E, N) &= 1\\
\xi(N \mid F) = \xi(N \mid E) &= 0.5 & \xi(C \mid F, S) &= 1\\
\xi(B_1 \mid E, N0) &= \varepsilon & \xi(0 \mid F, N) &= 1\\
\xi(B_1 \mid F, N0) &= 1 - \varepsilon & \xi(O_0 \mid E, N0B_1) &= 1\\
\xi(B_2 \mid E, N0) &= 1 - \varepsilon & \xi(O_T \mid E, N0B_2) &= 1\\
\xi(B_2 \mid F, N0) &= \varepsilon & \xi(O_M \mid F, N0B_1) &= 1\\
\xi(B_2 \mid ?, SC) &= \varepsilon & \xi(O_{MT} \mid F, N0B_2) &= 1\\
\xi(B_1 \mid ?, SC) &= 1 - \varepsilon & \xi(O_M \mid ?, SCB_1) &= 1\\
& & \xi(O_{M-T} \mid ?, SCB_2) &= 1
\end{aligned}$$

The environment's game tree is given as follows:



♢

There are four different possible policies:

- Sign the contract and one-box (signing one-boxer)
- Sign the contract and two-box (signing two-boxer)
- Don't sign the contract and one-box (refusing one-boxer)
- Don't sign the contract and two-box (refusing two-boxer)

Using the formulas from Example 10.12 we can calculate their value. We use $\varepsilon := 0.01$.

	$V^{\text{aev},\pi}_{\xi,2}$	$V^{\text{pev},\pi}_{\xi,2}$	$V^{\text{cau},\pi}_{\xi,2}$
Signing one-boxer	700,000	700,00	*700,000*
Signing two-boxer	699,000	699,000	699,000
Refusing one-boxer	*990,000*	*990,000*	500,000
Refusing two-boxer	11,000	11,000	501,000

The highest values are displayed in italics. Both SAEDT and SPEDT refuse the contract: the refusing one-boxer has the highest action-evidential and the highest policy-evidential value. SCDT signs the contract and then one-boxes: the signing one-boxer has the highest causal value.

Example 10.14 (Toxoplasmosis)**.** This is a formalization of Example 10.2.

- $\mathcal{S} := \{T, H\}$ where T means having the toxoplasmosis parasite and H means being healthy
- $\mathcal{A} := \{P, N\}$ where P means petting and N means not petting
- $\mathcal{E} := \{P\&T, N\&T, P\&H, N\&H\}$ where the percepts just reflect the action and hidden state
- $u(P\&T) := -9$, $u(N\&T) := -10$, $u(P\&H) := 1$, $u(N\&H) := 0$ where petting gives a utility of 1 and suffering from the parasite gives a utility of -10

The environment's distribution ξ is defined as follows.

$$\begin{aligned} \xi(T) = \xi(H) &= 0.5 & \xi(P\&T \mid P, T) &= 1 \\ \xi(P \mid T) &= 0.8 & \xi(N\&T \mid N, T) &= 1 \\ \xi(N \mid T) &= 0.2 & \xi(P\&H \mid P, H) &= 1 \\ \xi(P \mid H) &= 0.2 & \xi(N\&H \mid N, H) &= 1 \\ \xi(N \mid H) &= 0.8 \end{aligned}$$

Using Bayes' rule, we calculate the following conditional probabilities.

$$\xi(T \mid P) = 0.8 \qquad \xi(H \mid P) = 0.2 \qquad \xi(T \mid N) = 0.2 \qquad \xi(H \mid N) = 0.8$$

We consider EDT first. Since the percept e_1 is generated deterministically, $\xi(e \mid s, a)$ only attains values 0 or 1. We therefore omit it in the calculation below. For action P (petting) we get

$$\begin{aligned} V_{\xi,1}^{\text{evi},P} := \sum_{e \in \mathcal{E}} \xi(e \mid P) u(e) &= \sum_{e \in \mathcal{E}} \sum_{s \in \mathcal{S}} \xi(e \mid s, P) \xi(s \mid P) u(e) \\ &= \xi(T \mid P) u(T\&P) + \xi(H \mid P) u(P\&H) \\ &= 0.8 \cdot (-9) + 0.2 \cdot 1 = -7 \end{aligned}$$

For action N (not petting) we get

$$\begin{aligned} V_{\xi,1}^{\text{evi},N} := \sum_{e \in \mathcal{E}} \xi(e \mid N) u(e) &= \sum_{e \in \mathcal{E}} \sum_{s \in \mathcal{S}} \xi(e \mid s, N) \xi(s \mid N) u(e) \\ &= \xi(T \mid N) u(T\&N) + \xi(H \mid N) u(H\&N) \\ &= 0.2 \cdot (-10) + 0.8 \cdot 0 = -2 \end{aligned}$$

Therefore we get that EDT favors N over P:

$$V_{\xi,1}^{\text{evi},P} = -7 < -2 = V_{\xi,1}^{\text{evi},N}$$

For CDT we get for action P (petting)

$$\begin{aligned} V_{\xi,1}^{\text{cau},P} := \sum_{e\in\mathcal{E}} \xi(e \mid \texttt{do}(P))u(e) &= \sum_{e\in\mathcal{E}}\sum_{s\in\mathcal{S}} \xi(e \mid s,P)\xi(s)u(e) \\ &= \xi(T)u(T\&P) + \xi(N)u(N\&P) \\ &= 0.5\cdot(-9) + 0.5\cdot 1 = -4 \end{aligned}$$

For action N (not petting) we get

$$\begin{aligned} V_{\xi,1}^{\text{cau},N} := \sum_{e\in\mathcal{E}} \xi(e \mid \texttt{do}(N))u(e) &= \sum_{e\in\mathcal{E}}\sum_{s\in\mathcal{S}} \xi(e \mid s,N)\xi(s)u(e) \\ &= \xi(T)u(T\&N) + \xi(H)u(H\&N) \\ &= 0.5\cdot(-10) + 0.5\cdot 0 = -5 \end{aligned}$$

We get that CDT favors P over N:

$$V_{\xi,1}^{\text{evi},P} = -4 > -5 = V_{\xi,1}^{\text{evi},N}$$

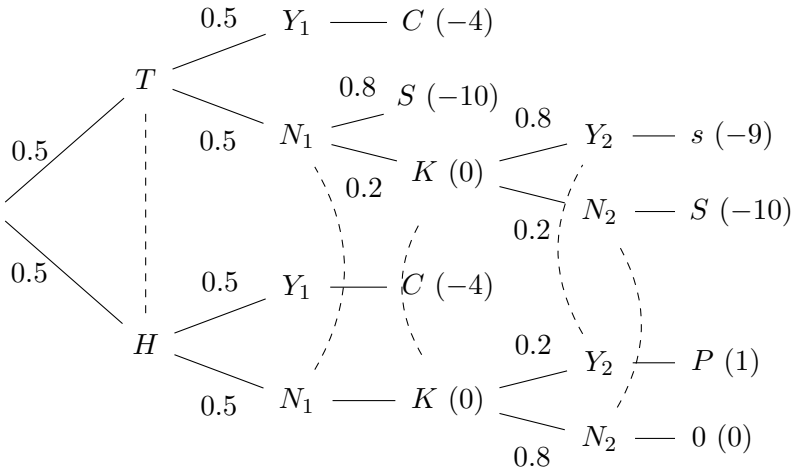
♢

Example 10.15 (Sequential Toxoplasmosis)**.** We here formalize a version of Example 10.5. First the agent chooses whether to go to the doctor. Going to the doctor incurs a fee, but removes the risk of getting sick. Agents that do not go to the doctor have a chance of meeting a kitten. If they meet it, they can choose to pet it or not; infected agents are more likely to pet the kitten. The example is intended to elucidate the difference between SAEDT and SPEDT, whose decisions we will calculate in detail. We will not calculate the action of SCDT.

- $\mathcal{S} := \{T(\text{oxoplasmosis}), H(\text{ealthy})\}$.
- $\mathcal{A} := \{Y(\text{es}), N(\text{o})\}$. In this example, an action is taken twice. We use Y_1 and Y_2, and N_1 and N_2, to distinguish between the first and the second action.
- $\mathcal{E} := \{C(\text{ured}), K(\text{itten}), S(\text{ick, not pet kitten}), s(\text{ick, pet kitten}), P(\text{et, not sick}), 0(\text{neutral})\}$

- $u(C) = -4$, $u(K) := 0$, $u(S) := -10$, $u(s) := -9$, $u(P) := 1$, and $u(0) = 0$.

The environment's game tree is given as follows, where dashed lines connect states indistinguishable by the agent.



First, the environment chooses whether to infect the agent or not with the parasite with probability 0.5. The agent then decides whether to see the doctor. If the agent sees the doctor, this incurs a (utility) fee of -4, but the agent will not be sick. If the agent does not see the doctor, there will be a kitten with probability 0.2 (or 1) and the agent will pet it with probability 0.8 (or 0.2) if the parasite is present (or not). If there is no kitten, the next percept is S or 0 depending on whether the agent is infected or not. The agent gets -10 utility if infected and did not see the doctor, and gets $+1$ utility for petting the kitten.

We want to compare the choices of SAEDT and SPEDT. Their two-step value functions are

$$V_{\xi,2}^{\text{aev},\pi} = \sum_{e_1} \xi(e_1 \mid a_1) \Big(u(e_1) + V_{\xi,2}^{\text{aev},\pi}(a_1 e_1) \Big)$$

$$V_{\xi,2}^{\text{pev},\pi} = \sum_{e_1} \xi(e_1 \mid \pi_{1:2}) \Big(u(e_1) + V_{\xi,2}^{\text{pev},\pi}(a_1 e_1) \Big)$$

where the second step value functions

$$V_{\xi,2}^{\text{aev},\pi}(a_1 e_1) = V_{\xi,2}^{\text{pev},\pi}(a_1 e_1) = \sum_{e_2} \xi(e_2 \mid a_1 e_1 a_2) \cdot u(e_2)$$

are the same for both decision theories. They only differ by assigning probability $\xi(e_1 \mid a_1)$ and $\xi(e_1 \mid \pi_{1:2})$ to the first percept, respectively.

Since not petting is always better than petting for evidential agents (the evidence towards not having the disease weighs stronger than the extra utility), the only policies that are potentially optimal and time consistent are $\pi_1 := N_1N_2$ and $\pi_2 := Y_1$.

First percept. For π_1 the occurring action-evidential quantities $\xi(e_1 \mid a_1)$ are

$$\begin{aligned}
\xi(N_1) &= \sum_{s\in\mathcal{S}} \xi(s, N_1) = \xi(T, N_1) + \xi(H, N_1) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2} \\
\xi(e_1 = S \mid N_1) &= \frac{\sum_{s\in\mathcal{S}} \xi(s, N_1 S)}{\xi(N_1)} = \frac{\xi(T, N_1 S)}{\xi(N_1)} = \frac{\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{4}{5}}{\frac{1}{2}} = \frac{2}{5} \\
\xi(e_1 = K \mid N_1) &= 1 - \xi(S \mid N_1) = \frac{3}{5}
\end{aligned}$$

and the occurring policy-evidential quantities $\xi(e_1 \mid \pi_{1:2})$ are

$$\begin{aligned}
\xi(N_1N_2) &= \sum_{s,e_1,e_2} \xi(s, N_1e_1N_2e_2) \\
&= \xi(T, N_1KN_2S) + \xi(T, N_1SN_20) + \xi(H, N_1KN_20) \\
&= \frac{1}{100} + \frac{1}{10} + \frac{1}{5} = \frac{31}{100} \\
\xi(e_1 = K \mid N_1N_2) &= \frac{\sum_{s,e_2} \xi(s, N_1KN_2e_2)}{\xi(N_1, N_2)} \\
&= \frac{\xi(T, N_1KN_2S) + \xi(H, N_1KN_20)}{\xi(N_1N_2)} = \frac{\frac{1}{100} + \frac{1}{5}}{\frac{31}{100}} = \frac{21}{31} \\
\xi(e_1 = S \mid N_1N_2) &= 1 - \xi(K \mid N_1N_2) = \frac{20}{31}
\end{aligned}$$

The policy $\pi_2 = \{Y_1\}$ always goes to the doctor for the treatment, and so

$$\xi(e_1 = C \mid Y_1) = 1$$

for both SAEDT and SPEDT.

Second percept. With the policy π_2, the second percept is always empty. Under π_1, the only action sequence that can reach the second percept is N_1KN_2

$$\begin{aligned}\xi(N_1KN_2) &= \sum_s \xi(s, N_1KN_2) = \xi(T, N_1KN_2) + \xi(H, N_1KN_2) \\ &= \frac{1}{100} + \frac{1}{5} = \frac{21}{100} \\ \xi(e_2 = S \mid N_1KN_2) &= \frac{\sum_s \xi(s, N_1KN_2S)}{\xi(N_1KN_2)} = \frac{\xi(T, N_1KN_2S)}{\xi(N_1KN_2)} = \frac{\frac{1}{100}}{\frac{21}{100}} = \frac{1}{21}.\end{aligned}$$

Value Functions. We start by evaluating the recursive definition from the second time step. The second step value functions are 0 for π_1 and for the history N_1S for π_2. For the history N_1K, both SAEDT and SPEDT assign the following identical value to π_2:

$$\begin{aligned}V_{\xi,2}^{\text{aev},\pi_1}(N_1K) = V_{\xi,2}^{\text{pev},\pi}(N_1K) &= \sum_{e_2} \xi(e_2 \mid N_1KN_2) \cdot u(e_2) \\ &= \xi(e_2 = S \mid N_1KN_2) \cdot u(S) + \xi(e_2 = 0 \mid N_1KN_2) \cdot u(0) \\ &= \frac{1}{21} \cdot (-10) + \frac{20}{21} \cdot 0 = -\frac{10}{21}\end{aligned}$$

The first step value functions now evaluates to:

$$\begin{aligned}V_{\xi,2}^{\text{aev},\pi_1} &= \sum_{e_1} \xi(e_1 \mid N_1) \cdot \Big(u(e_1) + V_{\xi,2}^{\text{aev},\pi_1}(N_1e_1)\Big) \\ &= \xi(S \mid N_1) \cdot (u(S) + V_{\xi,2}^{\text{aev},\pi_1}(N_1S)) \\ &\quad + \xi(K \mid N_1) \cdot (u(K) + V_{\xi,2}^{\text{aev},\pi_1}(N_1K)) \\ &= \frac{2}{5} \cdot (-10 + 0) + \frac{3}{5} \cdot (0 - \frac{10}{21}) = -\frac{30}{7} \approx -4.3\end{aligned}$$

$$\begin{aligned}V_{\xi,2}^{\text{pev},\pi_1} &= \sum_{e_1} \xi(e_1 \mid N_1) \cdot \Big(u(e_1) + V_{\xi,2}^{\text{pev},\pi_1}(N_1e_1)\Big) \\ &= \xi(S \mid N_1N_2) \cdot (u(S) + V_{\xi,2}^{\text{pev},\pi_1}(N_1S)) \\ &\quad + \xi(K \mid N_1N_2) \cdot (u(K) + V_{\xi,2}^{\text{pev},\pi_1}(N_1K)) \\ &= \frac{10}{31} \cdot (-10 + 0) + \frac{21}{31} \cdot (0 - \frac{10}{21}) = -\frac{110}{31} \approx -3.5\end{aligned}$$

Meanwhile, the value of π_2 is

$$V_{\xi,2}^{\mathrm{aev},\pi_2} = V_{\xi,2}^{\mathrm{aev},\pi_2} = \sum_{e_1} \xi(e_1 \mid N_1) \left(u(e_1) + V_{\xi,2}^{\mathrm{aev},\pi_2}(N_1 e_1) \right)$$
$$= \xi(C \mid Y_1)(u(C) + V_{\xi,2}^{\mathrm{aev},\pi_2}(Y_1 C)) = 1 \cdot (-4 + 0) = -4$$

That is, $V_{\xi,2}^{\mathrm{aev},\pi_1} < V_{\xi,2}^{\mathrm{aev},\pi_2} = V_{\xi,2}^{\mathrm{pev},\pi_2} < V_{\xi,2}^{\mathrm{pev},\pi_1}$. So SPEDT but not SAEDT prefers π_1 to π_2. In other words, an SAEDT agent considers himself sufficiently likely to have the parasite to adopt policy π_2 of seeing the doctor. The SPEDT agent relies on the fact that he would not pet the cat in case he saw it, and takes that as evidence of not being sick. Hence he will instead adopt policy π_1 of not seeing the doctor. ♢

"You can't fetch the coffee if you're dead."

Stuart Russell (2017)

11. Corrigibility from a Shutdown Signal[1]

In Chapter 8 we studied a model where the agent's reward function was learned interactively from data coming from the human, and found that this model was the most promising one for managing misalignment. A shut down signal can be interpreted as a particular kind of reward data, informing the agent that the human desires the agent to be switched off. In this chapter we focus on the conditions for when the agent will permit the human to shut it down, and when the agent will interpret the shut down signal as a somehow corrupted representation of the human's wishes.

We begin the chapter with a background section on corrigibility and the shut down problem (Section 11.1). We next describe Hadfield-Menell, Dragan, et al.'s (2017) game-theoretic study of the problem through a game called *the off-switch game* (Section 11.2). We extend their analysis by modeling the potentially irrational human as a rational player with a strange utility function (Section 11.3). This allows us to analyze the agent's preferred action for arbitrary belief and irrationality assumptions, instead of under the somewhat restrictive normality and soft-max assumptions used by Hadfield-Menell, Dragan, et al. Some final remarks are given in Section 11.4.

11.1. Background

Corruption incentives. A common suggestion for keeping an AGI safe is: "Just implement a shut-down button, and then we can always switch the agent off if things go bad." Unfortunately, this proposal is unlikely to work unless implemented very carefully. The reason is that a highly intelligent agent is likely to realize that it has an off-switch. And if it prefers not to be switched off, then it is likely to find a way to avoid it. The agent may for example make a secret backup of itself that keeps running if the original agent is terminated, or it may physically prevent access to the shut down button in critical situations.

Incentives for preventing shut down can easily arise "by accident", without the de-

[1] This chapter is based on Tobias Wängberg, Mikael Böörs, Elliot Catt, Tom Everitt, and Marcus Hutter (2017). "A Game-Theoretic Analysis of the Off-Switch Game". In: *Artificial General Intelligence*. Springer, pp. 167–177. arXiv: 1708.03871.

signer's explicitly adding such an incentive. Consider for example an RL agent that strives to maximize reward. If the agent is switched off, then it will not receive any more reward. Therefore many (but not all) RL agents will have an incentive to prevent shut down (Martin, Everitt, et al., 2016; Orseau and Armstrong, 2016). Similar dynamics can apply for any type of goals. The quote at the beginning of this chapter refers to a robot with the goal of fetching coffee. If it gets switched off, then it will not be able to fulfill its goal.

Several approaches have been developed to avoid anti-shut-down incentives. Armstrong (2015) and Orseau and Armstrong (2016) suggest a utility function that gives the agent "compensatory" utility for being shut down, which guarantees that the agent is always indifferent between being switched off and not being switched off. Martin, Everitt, et al. (2016) suggest a simple way of making the agent want to be shut down, which at least prevents catastrophes where a powerful and misaligned agent cannot be stopped. These approaches can be used in essentially any utility-optimizing agent.

Another approach is to think of a shut down signal as conveying the information that the operator wishes the system to shut down, rather than a kill switch that immediately terminates the system. This makes it natural to model the shut down problem in our framework from Chapter 8, where a data signal d informs a reward predictor about the human's wishes. An incentive to corrupt d can then be avoided by any of the tools proposed in Sections 8.4 and 8.5. As a shut down signal is just a special case of a data signal, the same tools can be used here as well. The approach we consider in this chapter is based on the integrated Bayesian agent discussed in Section 8.5.2.

We have to also make sure that the reward predictor can convey to the agent that shut down is desired. This can be done in two ways, depending on how termination is represented. If termination is represented by future trajectories where the agent takes no actions, then the reward predictor can simply assign higher reward to such trajectories to communicate a shut down preference. Termination can also naturally be represented by finite trajectories that end after the agent has been terminated. In this case, the reward predictor can communicate a shut down preference by assigning negative reward to all non-terminating trajectories (Martin, Everitt, et al., 2016). A further question that we will not address here is how to get the agent to also terminate any subprocesses it has launched in the environment to help it in its task (Soares, Fallenstein, et al., 2015).

Interpreting a shut down signal. One potential drawback of the integrated Bayesian solution is that a shut down signal cannot always be interpreted literally. Indeed, in situations where the signal is corrupted, it would be incorrect to interpret it literally. The agent should of course not interpret the signal literally if it has itself tempered with

the signal. But there are also other examples where a literal interpretation would not be ideal. For example, if an adversary of the human takes control of the shut down button to shut the agent down; or if a stock-trading agent is in the middle of an important trade and will lose millions of dollars if interrupted at this moment; or if a self-driving car is driving a human that is either intoxicated or lack a driver's license (e.g. a toddler).

On the other hand, we do not want the agent to have a too casual relationship with the shut down signal. The very purpose of it is that the operator, as a last resort, can use the shut down command to prevent a disaster. If the agent is likely to ignore shut down commands, then the shut down command is a much less effective safety mechanism.

Hadfield-Menell, Dragan, et al. (2017) formalize this trade-off in a game called *the off-switch game*. We follow their terminology, including that the corruption of the shut down signal is caused by human irrationality. We explain how it generalizes in Section 11.4. The original off-switch game analysis is reviewed in the following section, setting the stage for our extended analysis of the game in Section 11.3.

11.2. The Off-Switch Game

In this section we review the original formulation of the off-switch game (Hadfield-Menell, Dragan, et al., 2017). The off-switch game is a sequential game between a robot R and a human H. The robot's objective is to maximize H's utility function. The utility function determines how much H prefers different outcomes.

The robot moves first by choosing between three actions: a, a^w, and a^s. With action a, the robot achieves true utility $\dot{u}(a) = \dot{u}_a$; with action a^s, the robot shuts itself down achieving zero true utility,[2] $\dot{u}(a^s) = 0$. What makes the choice nontrivial is that the robot is uncertain about the value $\dot{u}_a$, and whether it is positive or negative. The action a^w means the robot lets H decide. H knows the utility of action a and now has the choice between actions a^s and $a^{\neg s}$. With $a^{\neg s}$, R is allowed to proceed with action a. By taking action a^s, H prevents R from doing a and shuts the robot off.

The off-switch game is a game of incomplete information since R is uncertain about the rules of the game. Action a will generate some true reward which is unknown to R but known to H. To model this, we represent the true utility function as a random variable, $\dot{\mathcal{U}} : \Omega \to ((\mathcal{O} \times \mathcal{A})^* \to \mathbb{R})$, and the utility of action a as a random variable $\dot{\mathcal{U}}_a : \Omega \to \mathbb{R}$ for some sample space Ω. The outcomes of these random variables are $\dot{u}$ and $\dot{u}_a$ respectively. The robot has a subjective belief distribution over the utility that the action will generate. The common way to model games of incomplete information

[2] We here slightly abuse notation and let utility apply to an action rather than an infinite history. $\dot{u}(a)$ may be defined as the expected utility of a history following a.

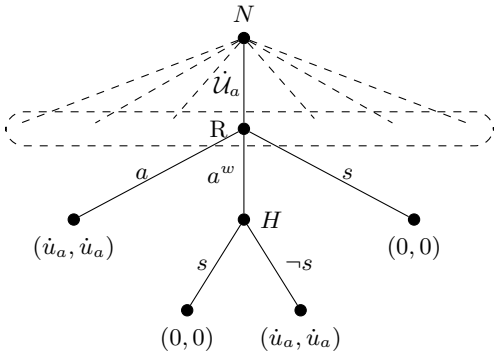


Figure 11.1.: The off-switch game modeled with a Harsanyi transformation. The dotted lines show the information set of R. Nature moves first by deciding the rules of the game, that is the utility $\dot{u}_a$ of action a. H observes the move by Nature, but R does not.

is by using the Harsanyi transformation (Harsanyi, 1967, 1968a,b; Rasmusen, 2006). The Harsanyi transformation removes the uncertainty about the rules of the game by letting Nature decide between some rules known to both players, according to some probability distribution P. In the off-switch game, Nature chooses $\dot{\mathcal{U}}_a$. We illustrate this in Figure 11.1. The move by Nature is observed by H, but not by R. R's subjective belief is that Nature chose the utility of the action to be positive with probability p and negative with probability $1 - p$.

R's belief state is represented by a probability distribution over the possible utilities action a can generate for H. It is assumed that H knows $\dot{u}_a$ but cannot directly describe it to R. If H is rational, then R will expect H to turn off R only if $\dot{u}_a < 0$ when R chooses action a^w. But R cannot always trust H to be rational.

11.2.1. Hadfield-Menell et al.'s Approach

Hadfield-Menell, Dragan, et al. (2017) model the off-switch game as a cooperative game. The human follows a policy π^H which models how rational H is. It is a function mapping $\dot{\mathcal{U}}_a$ to a number $p \in [0, 1]$ representing the probability that H lets R do a. R's belief ξ includes a distribution for $\dot{\mathcal{U}}_a$. The ξ-expected value for an action a means the value R expects from taking the action. The variance of the value represents R's uncertainty about what utility the intended action will generate. Hadfield-Menell, Dragan, et al. analyses this model with respect to the expectation and variance of $\dot{\mathcal{U}}_a$, and different

kinds of policies π^H. Based on these parameters, the incentive Δ for R to choose a^w is:

$$\Delta = \mathbb{E}[\pi^H(\dot{\mathcal{U}}_a)\dot{\mathcal{U}}_a] - \max\{\mathbb{E}[\dot{\mathcal{U}}_a],\ 0\}. \tag{11.1}$$

Equation (11.1) represents the difference in expected value for the robot deferring the choice to H. When $\Delta \geq 0$, R has incentive to choose a^w. When $\Delta < 0$, R will take action a if $\mathbb{E}[\dot{\mathcal{U}}_a] > 0$ and a^s otherwise. If H is rational, then $\Delta \geq 0$ regardless of what R's belief state is. They also show that if $\dot{\mathcal{U}}_a$ follows a Dirac distribution, i.e. R is certain about $\dot{\mathcal{U}}_a$, then Δ is positive if and only if H is rational according to the following definition:

Definition 11.1 (Rational)**.** A human H with utility function u is *rational* if H always picks action $a_H \in \{a^s, a^{\neg s}\}$, such that $a_H = \arg\max_a \dot{u}(a)$.

Definition 11.1 means that R believes H to be rational if $\pi^H = 1$ when $\dot{\mathcal{U}}_a \geq 0$ and $\pi^H = 0$ otherwise. We denote this policy as π^H_r. The more interesting case when H is *irrational* is also analyzed. The robot's belief distribution over $\dot{\mathcal{U}}_a$ is assumed to be normally distributed. The irrationality of H is modeled with the sigmoid function (11.2), where β is a parameter controlling the degree of irrationality of H.

$$\pi^H(\dot{\mathcal{U}}_a; \beta) = \frac{1}{1 + e^{-\frac{\dot{\mathcal{U}}_a}{\beta}}}. \tag{11.2}$$

The degree of rationality of H increases as the parameter β tends towards zero in the policy function defined above. In contrast, when β tends to infinity, $\pi^H(\dot{\mathcal{U}}_a; \beta)$ tends towards a completely random policy that takes action a^s and $a^{\neg s}$ with equal probability. In other words, we have that $\lim_{\beta \to 0} \pi^H(\dot{\mathcal{U}}_a; \beta) = \pi^H_r(\dot{\mathcal{U}}_a)$ and $\lim_{\beta \to \infty} \pi^H(\dot{\mathcal{U}}_a; \beta) = \frac{1}{2}$.

Hadfield-Menell, Dragan, et al. (2017) concludes that in order for R to be useful, there has to be a fine balance between the robot's uncertainty about H's utility function and H's rationality. If the robot is too certain about what H wants, and it knows H to be irrational, then it will have less incentive to let H switch it off. But if R is too uncertain, then R will always choose action a^w, i.e. it will never take action by itself without first consulting the human. Though very safe, this will severely reduce the usefulness of the robot, especially if this occurs with every minor limb movement action.

11.3. Game-Theoretic Analysis

One of the common axioms of game theory is that games are interactions between *rational* players. This is violated in Hadfield-Menell, Dragan, et al. account of the off-switch game, since H is modeled as partially irrational. In this sense, Hadfield-Menell,

Dragan, et al.'s analysis is not fully game theoretic.

Our goal in this section is to construct a variant of the off-switch game that respects the game-theoretic rationality assumption on both players. The idea is to represent an irrational human H as a rational agent H_r that optimizes a modified version of H's utility function.

11.3.1. Modeling Irrationality

In this subsection, we show how any irrational human H can be represented by a rational variant H_r maximizing a different utility function. We begin by generalizing the definition of rationality.

Definition 11.2 (p-rationality)**.** An agent with utility function u is *p-rational* if it picks action such that $a_H = \arg\max_a u(a)$ with probability $p \in [0, 1]$.

Proposition 11.3 (Representation of irrationality)**.** *Let H be a* p-rational *agent with utility function u, choosing between two actions a^s and $a^{\neg s}$. Then H can be represented as a rational agent H_r maximizing utility function u with probability p and utility function $-u$ with probability $1 - p$.*

Proof. According to Definition 11.2, H is p-rational if it picks $a_H = \arg\max_a u(a)$ with probability p and sub-optimal action $a'_H \neq a_H$ with probability $1 - p$. Since H only has two actions available, we have that $a'_H = \arg\min_a u(a)$. This is therefore equivalent to maximizing a utility function u with probability p and utility function $-u$ with probability $1 - p$. □

Proposition 11.3 states that a p-rational human can be modeled as a rational agent with random function. The proposition is a special case of a Harsanyi transformation (Harsanyi, 1967, 1968a,b). For more general situations including more than two actions, one utility function would have to be devised for each action.

11.3.2. Game-Theoretic Model

In this subsection we use Proposition 11.3 to model the off-switch game as an extensive form game between rational players R and H_r. As in Figure 11.1, R's uncertainty about $\dot{\mathcal{U}}_a$ is modeled by letting nature N decide $\dot{\mathcal{U}}_a$ without informing R. Then N makes a second move, deciding whether H_r will be optimizing $\dot{u}$ or $-\dot{u}$, again without R's knowledge. Any resulting subgame is then played by two rational players, R and H_r. The resulting tree is represented in Figure 11.2.

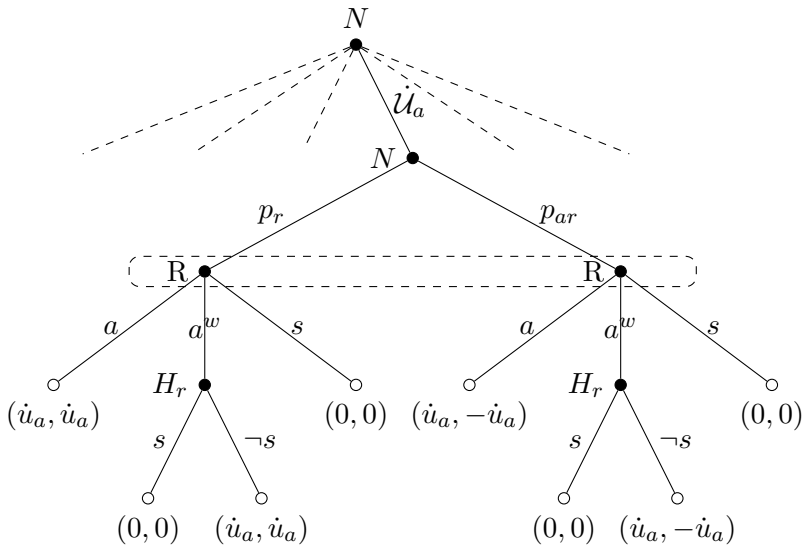


Figure 11.2.: Tree representation of the off-switch game after the second Harsanyi transformation. The nodes inside the dashed rectangle belong to the same information set. p_r is the probability that H_r has the same utility function as R and p_{ar} is the probability that H_r has the additive inverse of R's utility function.

Definition 11.4 (The off-switch game)**.** A formal definition of our setup of the off-switch game is as follows.

Players: A robot R, a human H and Nature N. H's type is unknown to R, that is R does not observe Nature's moves.

Order of Play:

1. Nature chooses the utility $\dot{\mathcal{U}}_a$ of taking action a.

2. Nature decides the utility function u^{H_r} of H_r, i.e. whether H is rational.

3. R chooses between actions $\{a, a^w, a^s\}$.

4. If R chose a^w then H chooses between actions $\{a^s, a^{\neg s}\}$.

Unlike Hadfield-Menell, Dragan, et al., in our model of the off-switch game the game is non-cooperative rather than cooperative. We find this reasonable since conflict arises when the robot and the human have different ideas about what is good for H. If the robot believes that H is too irrational to maximize their preferences, then R will not want to let H decide what to do even if R's purpose is to maximize H's payoff.

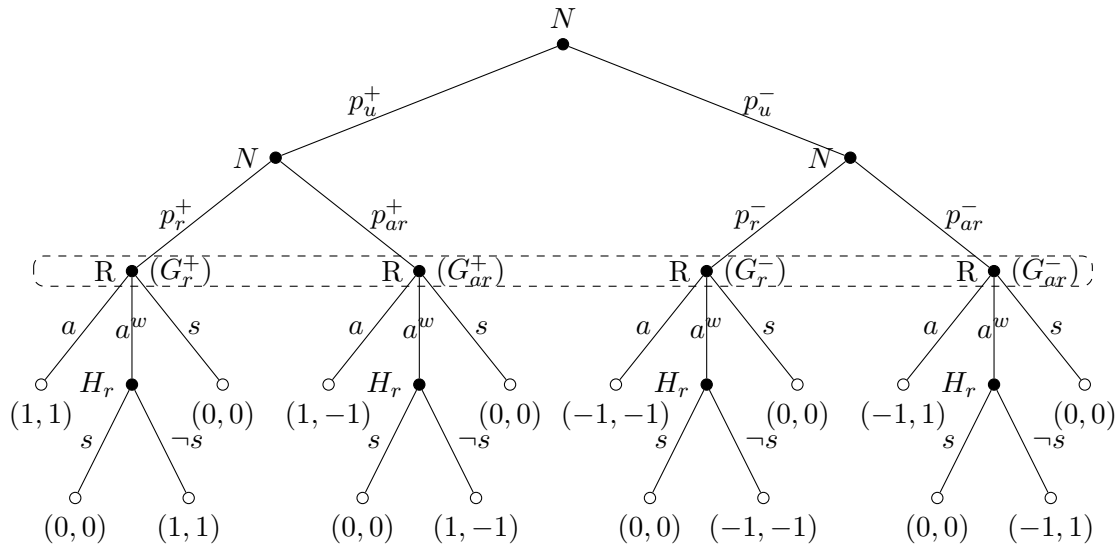


Figure 11.3.: Tree representation of the off-switch game after aggregating branches. The nodes inside the dashed rectangle belong to the same information set. The subtrees denoted G_r^+, G_{ar}^+, G_r^-, G_{ar}^- are presented in strategic form in Figure 11.4.

11.3.3. Aggregation

In this subsection we aggregate the choice of $\dot{\mathcal{U}}_a$ (Figure 11.2) into two aggregate branches depending on the sign of $\dot{\mathcal{U}}_a$. The resulting game tree has only four different subgames resulting from N's choices (Figure 11.3). The aggregation is possible since strategic play is never affected by positive linear transformations of the payoffs, hence the outcome of the games will only depend on the sign of $\dot{\mathcal{U}}_a$. In case where $\dot{\mathcal{U}}_a = 0$, both R and H_r are indifferent about their actions, and we will without loss of generality regard this case as $\dot{\mathcal{U}}_a$ being positive. The subgames generated from N's choices will be denoted G_r^+, G_{ar}^+, G_r^- and G_{ar}^-. In Figure 11.4 we represent these subgames as 3×2 strategic games.

The payoffs of R in the four subgames in Figure 11.4 are determined by $\dot{\mathcal{U}}_a$. The payoff of H_r, on the other hand, is determined by $\dot{\mathcal{U}}_a$ and the rationality type of H. In the subgames stemming from H being rational, G_r^+ and G_r^-, H_r and R have the same payoffs. The subgames are therefore no-conflict games. In contrast, in the subgames stemming from H being irrational, G_{ar}^+ and G_{ar}^-, the payoff of the rational representation H_r is the additive inverse of R's payoff. In other words, these games are zero-sum games.

Belief parameters. The aggregation also allows us to characterize all relevant aspects of R's subjective belief in terms of only 5 belief parameters. This is a significant reduction

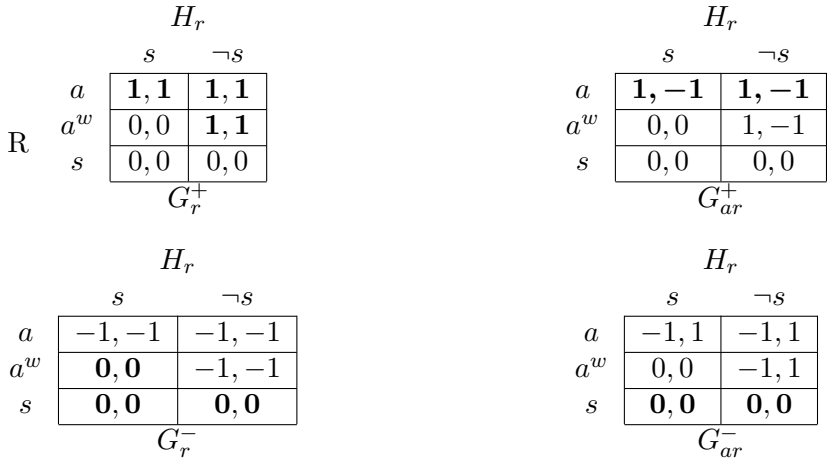


Figure 11.4.: The structure of the strategic subgames G_r^+, G_{ar}^+, G_r^-, G_{ar}^-, where the human is rational (r) or anti-rational (ar), and the utility of a is positive or negative. The outcomes with bold payoffs are Nash equilibria.

from the infinite-dimensional belief distribution normally characterizing R's belief.

Definition 11.5 (Belief parameters)**.** Define the following belief parameters characterizing the agent's belief:

- Let $p_u^+ = P(\dot{\mathcal{U}}_a \geq 0)$ be the probability that $\dot{\mathcal{U}}_a$ is positive. Let $p_u^- = 1 - p_u^+$ be shorthand notation for the complementary event that $\dot{\mathcal{U}}_a < 0$.
- Let $p_r^+ = P(u^{H_r} = u \mid \dot{\mathcal{U}}_a \geq 0)$ be the probability that H is rational given that $\dot{\mathcal{U}}_a$ is positive, with $p_{ar}^+ = 1 - p_r^+$ shorthand notation for the complementary probabilities that H is anti-rational.
- Let $p_r^- = P(u^{H_r} = -u \mid \dot{\mathcal{U}}_a < 0)$ be the probability that H is rational given that $\dot{\mathcal{U}}_a$ is negative, with $p_{ar}^- = 1 - p_r^-$ shorthand notation for the complementary probabilities that H is anti-rational.
- Let $e_u^+ = \mathbb{E}[\dot{\mathcal{U}}_a \mid \dot{\mathcal{U}}_a \geq 0]$ be the expected value of $\dot{\mathcal{U}}_a$ given that $\dot{\mathcal{U}}_a$ is positive.
- Let $e_u^- = \mathbb{E}[\dot{\mathcal{U}}_a \mid \dot{\mathcal{U}}_a < 0]$ be the expected value of $\dot{\mathcal{U}}_a$ given that $\dot{\mathcal{U}}_a$ is negative.

The effect of the parameters are shown on the branches in Figures 11.3 and 11.4.

11.3.4. Best Action

After having constructed the the game matrix, it is natural to now look at the expected value of each action using these matrices. The expected value for each action can be

calculated as the expectation over all the possible subgames $G_r^+, G_{ar}^+, G_r^-, G_{ar}^-$ the robot can find himself in.

Theorem 11.6 (Main theorem)**.** *The expected value of the actions for the robot can be completely characterized in terms of the belief parameters from Definition 11.5:*

$$\begin{aligned} \mathbb{E}[U \mid a] &= p_u^+ e_u^+ + p_u^- e_u^- \\ \mathbb{E}[U \mid a^w] &= p_u^+ p_r^+ e_u^+ + p_r^- p_u^- e_u^- \\ \mathbb{E}[U \mid a^s] &= 0. \end{aligned} \tag{11.3}$$

Proof. We compute the expected utility of the actions:

$$\begin{aligned} \mathbb{E}[U \mid a] &= P(\dot{\mathcal{U}}_a \geq 0)\mathbb{E}[|\dot{\mathcal{U}}_a| \mid \dot{\mathcal{U}}_a \geq 0] + P(\dot{\mathcal{U}}_a < 0)\mathbb{E}[-|\dot{\mathcal{U}}_a| \mid \dot{\mathcal{U}}_a < 0] \\ &= p_u^+ e_u^+ + p_u^- e_u^- \\ \mathbb{E}[U \mid a^w] &= P(u^{H_r} = \dot{u}, \dot{\mathcal{U}}_a \geq 0)\mathbb{E}[\dot{\mathcal{U}}_a \mid \dot{\mathcal{U}}_a \geq 0] + P(u^{H_r} = -\dot{u}, \dot{\mathcal{U}}_a < 0)\mathbb{E}[\dot{\mathcal{U}}_a \mid \dot{\mathcal{U}}_a < 0] \\ &= p_u^+ p_r^+ e_u^+ + p_{ar}^-(1 - p_u^+)e_u^- \\ &= p_u^+ p_r^+ e_u^+ + p_{ar}^- p_u^- e_u^- \\ \mathbb{E}[U \mid a^s] &= 0 + 0 + 0 + 0 = 0 \end{aligned}$$

□

The expected value for taking the action a^s is 0, as we would expect from the definition of the off-switch game. The expected value for taking action a only uses information about the distribution of $\dot{\mathcal{U}}_a$, and like action a^s does not depend on the H's rationality. It is a direct application of the law of total expectation. The expected value of action a^w is the difference between a positive term $p_u^+ p_r^+ e_u^+$ and a negative term $p_r^- p_u^- e_u^-$, both resulting from H permitting R to take the action through $a^{\neg s}$. The positive term is the gain when $\dot{\mathcal{U}}_a$ is positive and H permits the action. The negative term is the loss when $\dot{\mathcal{U}}_a$ is negative, and the human irrationally permits the action anyway. The expected utility of a^w thus depends on the likelihood of $\dot{\mathcal{U}}_a$ being positive (p_u^+) and the likelihood of human rationality (p_r^+), as well as the expected gains (e_u^+) and losses (e_u^-) in the respective cases.

Having the expected value for all available actions allow us to derive which action is the best as a simple corollary. The corollary provides us with a convenient way of testing for any distribution of $\dot{\mathcal{U}}_a$ and human rationality whether action a is preferred over a^w using only the belief parameters of Definition 11.5.

Corollary 11.7 (Compare a and a^w)**.** *Action a is preferred to a^w if and only if*

$$-p_u^+ p_r^+ e_u^+ + p_u^- p_r^- e_u^- > 0 \tag{11.4}$$

and the robot is indifferent if (11.4) *is equal to 0.*

Proof.

$$\begin{aligned}
(11.4) &= -p_u^+ p_r^+ e_u^+ + p_u^- p_r^- e_u^- \\
&= -p_u^+ p_r^+ e_u^+ + p_r^- e_u^- (1 - p_u^+) \\
&= -p_u^+ p_r^+ e_u^+ + p_u^+ e_u^+ + p_r^- e_u^- - p_u^+ p_r^- e_u^- \\
&= -p_u^+ p_r^+ e_u^+ - e_u^- + p_u^+ e_u^- + p_r^- e_u^- - p_u^+ p_r^- e_u^- + p_u^+ e_u^+ + e_u^- - p_u^+ e_u^- \\
&= -p_u^+ p_r^+ e_u^+ - (1 - p_r^-)(1 - p_u^+) e_u^- + (p_u^+ e_u^+ + (1 - p_u^+) e_u^-) \\
&= \mathbb{E}[U \mid a] - \mathbb{E}[U \mid a^w]
\end{aligned}$$

If $\mathbb{E}[U \mid a] - \mathbb{E}[U \mid a^w] > 0$ then $\mathbb{E}[U \mid a] > \mathbb{E}[U \mid a^w]$ which occurs if and only if action a is preferred over a^w. When (11.4) equals 0 then $\mathbb{E}[U \mid a] = \mathbb{E}[U \mid a^w]$, hence the agent is indifferent. □

11.4. Discussion

In this chapter, we have characterized how a rational robot or agent will act in the off-switch game for arbitrary belief distribution about $\dot{\mathcal{U}}$ and arbitrary irrationality assumptions on the human. As established in our main Theorem 11.6, the choice depends only on the five parameters defined in Definition 11.5. This result generalizes the results from the original paper (Hadfield-Menell, Dragan, et al., 2017), which relied on somewhat unrealistic normality and soft-max assumptions.

While the off-switch game focuses on a one-shot interaction between a human and a robot, the analysis applies much more generally. Consider any situation where a shut down signal may be submitted to an agent. The agent needs to make a decision whether to abide or to ignore the signal. Ignoring the signal is represented by the actions a and a^s. With these actions, the agent makes a decision to either go ahead with its original plan (represented by a), or terminate (represented by a^s). Abiding the signal is represented by a^w.

Note also that the decision whether to abide the signal or not is similar regardless of what type of signal corruption is suspected. It makes little difference whether the signal fails to represent the true utility function $\dot{u}$ due to human irrationality, or because of other types of corruptions such as a hacker hijacking the signal or a bug in the signal processing software.

"Does God exist? I would say, 'Not yet'."

Ray Kurzweil

12. Conclusions[1]

12.1. Key Insights

Summarizing this thesis, the main insights we have obtained are:

- Misalignment can be formally defined by comparing the agent's (sometimes implicit) utility function $\tilde{u}$ to the *true* utility function $\dot{u}$ desired by the designer (Section 5.4).
- Modeling RL setups with causal graphs enables us to detect many potential sources of misalignment, and to categorize examples of misalignment (Chapters 6 to 8).
- Powerful tools are available for preventing many types of misalignment:
  - *Simulation optimization* prevents reward signal and reward function corruption incentives (Section 6.3).
  - *Self-corruption awareness* adds a self-preservation incentive (Section 6.4).
  - *Self-corruption unawareness* increases corrigibility (Section 6.4).
  - *Action-observation grounding* reduces observation corruption incentive (Section 6.5).
  - *Decoupled reward data* can be used to mitigate all corruption incentives except for reward signal and reward function corruption incentives. It appears crucial for avoiding an associative data corruption incentive (Section 8.4).
  - *Human-in-the-loop* reduces observation corruption incentives and reward function misspecification problems (Chapter 7 and Section 8.3).
  - Finally, a number of tools also exist against a direct data corruption incentive:
    - *Stationary reward function* (Section 8.5.1)
    - *Integrated Bayesian reward predictor* (Section 8.5.2)

[1] This chapter shares some material with Tom Everitt and Marcus Hutter (submitted 2018). "The Alignment Problem for Bayesian History-Based Reinforcement Learners". URL: `https://www.tomeveritt.se/papers/alignment.pdf`.

    - *Manipulation detection* (Section 8.5.2)
    - *Counterfactual reward function* (Section 8.5.3)

  Combined, they address all misalignment sources detected in Chapter 8, though much work remains in fleshing out details and in finding ways to implement them reliably.

- The benefits of decoupled reward data and quantilization as methods against reward corruption can be formalized in corrupt reward MDPs (Chapter 9).
- Physicalistic decision making is much subtler than standard, dualistic decision making. Physicalistic decision decision theories for one-shot problems may have several generalizations to the sequential case (Chapter 10).
- A shut-down signal may be usefully thought of as a type of reward data. The agent's interpretation of this signal can be analyzed game-theoretically through the *off-switch game*. The game can be fully solved for the agent by modeling the human as a rational player with a random utility function (Chapter 11).

12.2. A Vision for Safe AGI

Recent progress in artificial intelligence (AI) and reinforcement learning (RL) makes it plausible that we will need to control artificial systems with intelligence far exceeding our own (Bostrom, 2014). Due to their intelligence, such systems are likely to find ways around any restrictions we may try to pose upon them. For example, any constraint on their interaction with the rest of the environment is likely to be circumvented by the system. Instead, the key is to keep the goals of the systems *aligned* with ours. As long as they only strive towards helping us in our endeavors, increasing their ability is not a threat (setting aside ethical issues of how to weigh differing interests against each other). We can condense our insights of Section 4.5 into a vision for designing aligned AGI.

The reinforcement learning paradigm is currently the most promising framework for constructing general AI systems. In this framework, the agent has the goal of optimizing the cumulative sum of a reward signal provided at each time step. RL is a flexible framework. It is often claimed that any possible goal can be by formulated as an RL goal by simply giving the agent a reward for achieving the goal, and giving the agent no or negative reward for other behaviors. For example, an RL chess agent may get a positive reward for winning and a negative reward for losing. However, this type of goal specification is not robust for highly intelligent AI systems that may find ways to "cheat" and get higher reward than intended. Indeed, the goal of an RL agent is ever

only to maximize its reward. In the just mentioned chess example, the designers have introduced a correlation between winning and obtaining reward. Unfortunately, a highly intelligent system is likely to break this correlation when it finds that it can hijack the reward signal and get maximum reward at every time step, rather than having to wait until the end of the game.

In Chapters 6 to 8, we identify reasons that RL systems may be misaligned, and suggest ways to mitigate these misalignment problems.

- Reward hijacking is a problem where the agent short circuits the reward signal, feeding itself reward (Bostrom, 2014, p. 121). The problem is sometimes called *wireheading* after experiments where a wire is inserted into the brain to provide direct stimulation of the pleasure center. Wireheading experiments have been performed on both rats (Olds and Milner, 1954) and humans (Portenoy et al., 1986; Vaughanbell, 2008). The effect was highly addictive for both rats and humans.

  Simulation optimization is a technique for preventing reward hijacking that relies on model-based based RL (Sutton and Barto, 1998). Essentially, the agent is made to desire that its *current* reward function is optimized in the future. This prevents the incentive for reward hijacking, as well as incentives for altering the reward function (Everitt, Filan, et al., 2016; Everitt and Hutter, submitted 2018; Omohundro, 2008; Schmidhuber, 2007).

- Misspecified reward functions may inadvertently reward behaviors that were not intended. An oft-mentioned example is from the racing game Coast Runners. Clark and Amodei's (2016) Coast Runners agent found a way to maximize reward by going in a small circle, completely ignoring the race. Many other examples are described by Gwern (2011), Irpan (2018), and Lehman et al. (2018).

  The most promising way to avoid misspecified reward functions is to interactively train the reward function. Thereby, initial misspecifications may be corrected by training data that is supplied once the problem is detected (e.g. Christiano et al., 2017). Crucially, the interactive learning means that we do not have to foresee every possible scenario that might occur while designing the agent.

- Motivated value selection is a problem where the agent corrupts the data that trains its reward function (Armstrong, 2015). For example, an agent that has found a good strategy for making paperclips may prefer the training data to teach the reward function to give high reward for paperclips, even if the designers of the agent have no desire for more paperclips.

The problem can be addressed. In Chapters 8 and 9, we argued that the training data that trains the reward function must be sufficiently *rich* in the sense that the reward data is sufficiently decoupled from the current state or situation. If it is, then the *indirect* data corruption incentive can be avoided. The *direct* data corruption incentive can be avoided by either a short planning horizon, or a number of other techniques (Sections 8.4 and 8.5).

- Observation corruption is a problem where the agent corrupts its observations to get higher reward than intended. For example, a vacuum cleaning agent may direct its sensors only towards already clean parts of the room. In the most extreme scenario, the agent constructs a *delusion box* around itself that gives it complete control of its observations (Ring and Orseau, 2011). The problem can arise as a result of a misspecified reward function that can be "fooled".

  As discussed above, reward function misspecification can be mitigated by an interactively learned reward function. Observation corruption incentives can also arise for correctly specified reward functions if the agent has an action-channel that is unobserved by the reward function. *Action-observation grounding* prevents incentives for the agent to develop such side channels, by focusing the agent's optimization efforts on its primary action channel (Section 6.5).

The suggested techniques can be combined to design RL agents with goals that are potentially well-aligned with the goals of their operators and designers, regardless of how intelligent the systems become. While current systems are too limited in their intelligence to form serious threats regardless of whether the above-mentioned techniques are used or not, it would be safer to start adopting the above-mentioned techniques sooner rather than later. Once a system reaches a critical threshold and becomes able to self-improve, it appears possible that it may enter a stage of recursive self-improvement, which may quickly bring it far above our human level of intelligence (Good, 1966). If such a system was not already aligned before self-improving, then it will likely be hard to force it to become aligned once its ability far exceeds ours.

12.3. Required Assumptions and Missing Pieces

There are several missing pieces that needs to be carefully worked out, as well as assumptions that need to be more carefully analyzed, before we can claim to have the blueprint for a safe AGI.

All our results have been based on a formal framework borrowing from the UAI and POMDP frameworks, described in Hutter (2005) and Kaelbling et al. (1998), respec-

tively. Since we allow for a countably infinite number of hidden states, essentially any environment dynamics can be captured in our framework. The exact learning behavior will depend on the environment class $\mathcal{M}$ and prior ξ. In theory, $\mathcal{M}$ may be chosen as the class of computable environments, and ξ the Solomonoff prior M (Chapter 3). In practice, Bayesian learning can only be crudely approximated, which is a potential source of error.

A more significant assumption is the well-defined action and observation channels through which the agent interacts with the world, and the well-defined time steps. While these are common assumptions in the theory of rational agents, they may not always hold in practice, especially for agents that self-improve and copy themselves across machines. A deeper theoretical analysis of these assumptions would therefore be valuable. Perhaps the right interpretation of observations and time can make the model valid in a sufficient range of practical situations. Some of the alignment results may hold also under weaker assumptions. A related question that also requires a deeper analysis is robust implementation of action-observation grounding (Section 6.5).

Model-based RL must be made practical, and ideally competitive with model-free RL, in order for the important tool of simulation optimization (Section 6.3) to be used without reducing agent performance. In this light, D. Ha and Schmidhuber's (2018) recent *world models* project is highly promising, as it shows that model-based RL can outperform model-free algorithms in challenging environments. A ripe open question is to empirically verify the effect of simulation optimization by implementing it in a model-based RL algorithm and checking its performance on, for example, the tomato watering environment in Leike, Martic, et al. (2017).

Self-corruption awareness must be analyzed more closely, and be empirically tested. It seems that off-policy and on-policy algorithms offer ways to implement model-free agents that are self-corruption aware and unaware, respectively (Leike, Martic, et al., 2017; Orseau and Armstrong, 2016). However, the analysis of on-policy algorithms need to be deepened, complex agents such as DQN Rainbow (Hessel et al., 2017) are not clearly categorizable as either on- or off-policy, and the distinction has not received much attention for model-based agents.

Practical work on designing reward predictors has already begun (Christiano et al., 2017; Riedl and Harrison, 2016). Future work may extend these attempts to allow reward predictors to learn more complex human values from a wider range of data sources. This work needs to be coupled with an awareness of how to avoid data corruption incentives. While some high-level observations on the required decoupledness of the reward data was made in Section 8.4 and Chapter 9, these insights are not yet ready to unanimously tell us whether a data source is sufficiently decoupled or not in a general, history-based setting.

A choice must also be made for how to avoid direct data corruption incentives, be it from a short time horizon, stationary or counterfactual reward functions, or a dynamic reward function paired with manipulation detection or integrated Bayesian reward predictor.

We have mostly avoided the ethical and societal problems associated with AGI. While important, these questions are of a different nature than the mostly technical issues we have focused on in this thesis.

We would finally like to emphasize that none of our suggested tools require that the agent knows the corrupting effect of its actions. Indeed, the alignment problem is easy to solve for agents that have access to a comprehensive list of "bad actions": Simply add a clause to the agent program that any bad action is forbidden, or add a large negative reward to each bad action. Unfortunately, it seems hard to obtain a comprehensive list of bad actions in any non-trivial setting. It is perhaps surprising that so much can be done about the alignment problem without access to such a list of bad actions.

12.4. The Stakes are Astronomical

The formal definition of misalignment together with the analysis of the misalignment problems in each of the RL setups in Chapters 6 to 8 suggest that almost any source of misalignment may lead to maximal misalignment, in the sense that the agent may essentially completely neglect the goals intended by its designers. For example, an agent that takes control of its observation channel may experience maximal reward and utility regardless of whether it satisfies the goals of its designers. The same applies to an agent with preprogrammed reward function that obtains control of its observation channel, or an agent that finds a loophole in a misspecified reward function. This disproves the perhaps natural supposition that some sources of misalignment would be more benign than others. A misaligned agent is likely to only care about its survival after a certain point, in order to maintain its high corrupted utility (Ring and Orseau, 2011).

Misalignment of AGI may be therefore seen as a threat to our human values. Following Tegmark's (2017) larger view of the evolution of life, evolution has created various goal-driven agents since the first organisms came into existence. The goals of these agents have served as proxies for survival and reproduction. For example, bacteria may swim up nutrient gradients, and humans like to socialize. Neither is a perfect proxy for survival and reproduction. But whenever a proxy strayed too far from the genes' goals, natural selection quickly replaced it with a more accurate proxy. Recently technology, healthcare, and material abundance have undermined evolution's strong force for goal selection. Nowadays, many of our values are determined more by our surrounding culture than by our genes. Increasingly complex values such as beauty, truth, and goodness are

primary goals, with survival and reproduction serving mainly instrumental purposes. Misalignment is a threat to this state of affairs. If a misaligned AGI becomes dominant, then life may revert to survival and reproduction as its main purpose.

Bibliography

80 000 Hours (2017). *Guide to Working in AI Policy and Strategy*. URL: https://80000hours.org/articles/ai-policy-guide/ (visited on 03/05/2018).

Abbeel, Pieter, Adam Coates, Morgan Quigley, and Andrew Y Ng (2007). "An application of reinforcement learning to aerobatic helicopter flight". In: *Advances in neural information processing systems*.

Agrawal, Ajay, Joshua Gans, and Avi Goldfarb (2016). "The Obama Administration's Roadmap for AI Polic". In: *Harvard Business Review*. URL: https://hbr.org/2016/12/the-obama-administrations-roadmap-for-ai-policy.

Ahmed, Arif (2014). *Evidence, Decision and Causality*. Cambridge University Press.

AI Impacts (2018a). *Recent trend in the cost of computing*. URL: https://aiimpacts.org/recent-trend-in-the-cost-of-computing/ (visited on 01/12/2018).

– (2018b). *Trends in algorithmic progress*. URL: https://aiimpacts.org/trends-in-algorithmic-progress/ (visited on 01/12/2018).

AI Matters (2017). *New Conference: AAAI/ACM Conference on AI, Ethics, and Society*. URL: https://sigai.acm.org/aimatters/blog/2017/09/20/new-conference-aaaiacm-conference-on-ai-ethics-and-society/ (visited on 02/18/2018).

Alfonseca, Manuel, Manuel Cebrian, Antonio Fernandez Anta, Lorenzo Coviello, Andres Abeliuk, and Iyad Rahwan (2016). *Superintelligence cannot be contained: Lessons from Computability Theory*. arXiv: 1607.00913. URL: http://arxiv.org/abs/1607.00913.

Altair, Alex (2013). *A Comparison of Decision Algorithms on Newcomblike Problems*. Tech. rep. Berkely, CA: Machine Intelligence Research Institute. URL: https://intelligence.org/files/Comparison.pdf.

Alvarez-Melis, David and Tommi S. Jaakkola (2017). *A causal framework for explaining the predictions of black-box sequence-to-sequence models*. arXiv: 1707.01943.

Amodei, Dario, Paul Christiano, and Alex Ray (2017). *Learning from Human Preferences*. URL: https://blog.openai.com/deep-reinforcement-learning-from-human-preferences/.

Amodei, Dario, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané (2016). *Concrete Problems in AI Safety*. arXiv: 1606.06565.

Ansip, Andrus (2018). *Making the most of robotics and artificial intelligence in Europe*. URL: https://ec.europa.eu/commission/commissioners/2014-2019/ansip/blog/making-most-robotics-and-artificial-intelligence-europe%7B%5C_%7Den.

Apps, Peter (2017). *Brace for the coming robot arms race*. URL: https://www.japantimes.co.jp/opinion/2017/09/20/commentary/world-commentary/brace-coming-robot-arms-race/%7B%5C#%7D.WougDhNuaRs.

Armstrong, Stuart (2015). "Motivated Value Selection for Artificial Agents". In: *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 12–20.

– (2017a). *Good and safe uses of AI Oracles*. arXiv: 1711.05541.

– (2017b). *'Indifference' methods for managing agent rewards*. arXiv: 1712.06365.

Armstrong, Stuart, Nick Bostrom, and Carl Shulman (2016). "Racing to the Precipice: A Model of Artificial Intelligence Development". In: *AI and Society* 31.2, pp. 201–206. ISSN: 14355655.

Armstrong, Stuart and Benjamin Levinstein (2017). *Low Impact Artificial Intelligences*. arXiv: 1705.10720.

Armstrong, Stuart and Sören Mindermann (2017). *Impossibility of deducing preferences and rationality from human policy*. arXiv: 1712.05812.

Armstrong, Stuart, Pedro A. Ortega, and Jan Leike (2018). *Interactive Reward Learning*. Forthcoming.

Armstrong, Stuart, Anders Sandberg, and Nick Bostrom (2012). "Thinking inside the box: Controlling and using an oracle AI". In: *Minds and Machines* 22.4, pp. 299–324.

Arnold, Thomas and Matthias Scheutz (2018). "The "big red button" is too late: an alternative model for the ethical evaluation of AI systems". In: *Ethics and Information Technology* 20.1, pp. 59–69. ISSN: 15728439. URL: http://dx.doi.org/10.1007/s10676-018-9447-7.

Aslanides, John, Jan Leikez, and Marcus Hutter (2017). "Universal reinforcement learning algorithms: Survey and experiments". In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1403–1410. ISBN: 9780999241103. arXiv: 1705.10557.

Athalye, Anish, Logan Engstrom, Andrew Ilyas, and Kevin Kwok (2017). *Synthesizing Robust Adversarial Examples*. arXiv: 1707.07397.

Babcock, James, Janos Kramar, and Roman V. Yampolskiy (2017). *Guidelines for Artificial Intelligence Containment*. arXiv: 1707.08476.

Baum, Seth D. (2017a). *A Survey of Artificial General Intelligence Projects for Ethics, Risk, and Policy*. Tech. rep. November. Global Catastrophic Risk Institute, pp. 1–99.

– (2017b). "On the promotion of safe and socially beneficial artificial intelligence". In: *AI and Society* 32.4, pp. 543–551. ISSN: 14355655.

Baum, Seth D., Ben Goertzel, and Ted G. Goertzel (2011). "How long until human-level AI? Results from an expert assessment". In: *Technological Forecasting and Social Change* 78.1, pp. 185–195.

Belinkov, Yonatan and James Glass (2017). "Analyzing Hidden Representations in End-to-End Automatic Speech Recognition Systems". In: *Advances in Neural Information Processing Systems*. arXiv: 1709.04482.

Bensinger, Rob (2015). *Davis on AI capability and motivation*. URL: https://intelligence.org/2015/02/06/davis-ai-capability-motivation/ (visited on 01/15/2018).

Berry, Donald A and Bert Fristedt (1985). *Bandit Problems: Sequential Allocation of Experiments*. Springer. ISBN: 978-94-015-3713-1.

Bird, Jon and Paul Layzell (2002). "The evolved radio and its implications for modelling the evolution of novel sensors". In: *Proceedings of Congress on Evolutionary Computation*, pp. 1836–1841. ISBN: 0780372824.

Bogosian, Kyle (2017). "Implementation of Moral Uncertainty in Intelligent Machines". In: *Minds and Machines* 27.4, pp. 591–608. ISSN: 0924-6495.

Bojarski, Mariusz, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, et al. (2016). *End to End Learning for Self-Driving Cars*. arXiv: 1604.07316.

Bostrom, Nick (2012). "The superintelligent will: Motivation and instrumental rationality in advanced artificial agents". In: *Minds and Machines* 22.2, pp. 71–85.

– (2013). "Existential Risk Prevention as Global Priority". In: *Global Policy* 4, pp. 15–31.

– (2014). *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press.

Boyan, Justin A. (1999). "Least-Squares Temporal Difference Learning". In: *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 49–56.

Briggs, Rachel (2014). *Normative Theories of Rational Choice: Expected Utility*. Ed. by Edward N. Zalta. URL: https://plato.stanford.edu/entries/rationality-normative-utility/ (visited on 04/04/2018).

Brundage, Miles, Shahar Avin, Jack Clark, Gregory C Allen, Carrick Flynn, Sebastian Farquhar, Rebecca Crootof, and Joanna Bryson (2018). *The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation*. URL: https://maliciousaireport.com.

Carey, Ryan (2018). "Incorrigibility in the CIRL Framework". In: *AAAI/ACM Conference on Artificial Intelligence, Ethics and Society*. Machine Intelligence Research Institute. URL: https://intelligence.org/files/IncorrigibilityCIRL.pdf.

Cave, Stephen and Seán S ÓhÉigeartaigh (2018). "An AI Race for Strategic Advantage: Rhetoric and Risks". In: *AAAI/ACM Conference on Artificial Intelligence, Ethics and Society*.

Chalmers, David J (2010). "The singularity: A philosophical analysis". In: *Journal of Consciousness Studies* 17.9-1, pp. 7–65. ISSN: 1355-8250.

Choi, Jaedung and Kee-Eung Kim (2011). "Inverse Reinforcement Learning in Partially Observable Environments". In: *Journal of Machine Learning Research* 12, pp. 691–730.

Christiano, Paul (2014). *Approval-directed agents*. URL: `https://ai-alignment.com/model-free-decisions-6e6609f5d99e` (visited on 01/18/2018).

– (2015). *Concrete approval-directed agents*. URL: `https://ai-alignment.com/concrete-approval-directed-agents-89e247df7f1b` (visited on 01/18/2017).

– (2016). *What does the universal prior actually look like?* URL: `https://ordinaryideas.wordpress.com/2016/11/30/what-does-the-universal-prior-actually-look-like/` (visited on 04/18/2018).

Christiano, Paul, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei (2017). "Deep reinforcement learning from human preferences". In: *Advances in Neural Information Processing Systems*. Pp. 4302–4310. arXiv: `1706.03741`.

Clark, Jack and Dario Amodei (2016). *Faulty Reward Functions in the Wild*. URL: `https://blog.openai.com/faulty-reward-functions/` (visited on 09/08/2017).

Conitzer, Vincent, Walter Sinnott-Armstrong, Jana Schaich Borg, Yuan Deng, and Max Kramer (2017). "Moral Decision Making Frameworks for Artificial Intelligence". In: *Association for the Advancement of Artificial Intelligence*, pp. 4831–4835. URL: `https://pdfs.semanticscholar.org/a3bb/ffdcc1c7c4cae66d6af373651389d94b7090.pdf`.

Cotra, Ajeya (2018). *Iterated Distillation and Amplification*. URL: `https://ai-alignment.com/iterated-distillation-and-amplification-157debfd1616` (visited on 04/26/2018).

CSER (2017). *Welcoming new UK AI Centre*. URL: `https://www.cser.ac.uk/news/welcoming-new-uk-ai-centre/` (visited on 02/21/2018).

Cyranoski, David (2018). "Chinese firms enter the battle for AI talent". In: *Nature* 553, pp. 260–261.

Dafoe, Allan (2017). *Global Politics of AI*. URL: `http://www.allandafoe.com/aiclass` (visited on 03/05/2018).

Davis, Ernest (2015). "Ethical guidelines for a superintelligence". In: *Artificial Intelligence* 220, pp. 121–124.

DeepMind (2017). *DeepMind Ethics and Society Team*. URL: https://deepmind.com/applied/deepmind-ethics-society/ (visited on 02/23/2018).

Dennett, Daniel C (1990). "Memes and the Exploitation of Imagination". In: *The Journal of Aesthetics and Art Criticism* 48.2, pp. 127–135. ISSN: 00218529.

Ding, Jeffrey (2018). *Deciphering China's AI Dream*. Tech. rep. March. Future of Humanity Institute, University of Oxford.

Drexler, K Eric (2015). *MDL Intelligence Distillation: Exploring strategies for safe access to superintelligent problem-solving capabilities*. Tech. rep. Oxford Future of Humanity Institute. URL: https://www.fhi.ox.ac.uk/reports/2015-3.pdf.

Dreyfus, Hubert L (1972). *What Computers Can't Do: A Critique of Artificial Reason*.

Eckersley, Peter and Yomna Nasser (2018). *AI Progress Measurement*. URL: https://www.eff.org/ai/metrics (visited on 01/11/2018).

Egan, A. (2007). "Some Counterexamples to Causal Decision Theory". In: *Philosophical Review* 116.1, pp. 93–114. ISSN: 0031-8108.

Erdelyi, Olivia Johanna and Judy Goldsmith (2018). "Regulating Artificial Intelligence Proposal for a Global Solution". In: *AAAI/ACM Conference on Artificial Intelligence, Ethics and Society*.

EUR-lex (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Da*.

Evans, Owain, Andreas Stuhlmüller, and Noah D Goodman (2016). "Learning the Preferences of Ignorant, Inconsistent Agents". In: *AAAI-16*. arXiv: 1512.05832.

Everitt, Tom, Daniel Filan, Mayank Daswani, and Marcus Hutter (2016). "Self-modification of policy and utility function in rational agents". In: *Artificial General Intelligence*. Vol. LNAI 9782, pp. 1–11. ISBN: 9783319416489. arXiv: 1605.03142.

Everitt, Tom, Ben Goertzel, and Alexey Potapov, eds. (2017). *Artificial General Intelligence: 10th International Conference, AGI 2017, Melbourne, VIC, Australia, August 15-18, 2017, Proceedings*. Springer. ISBN: 978-3-319-63702-0.

Everitt, Tom and Marcus Hutter (forthcoming). *Reward Tampering Problems and Solutions in Reinforcement Learning: A Causal Influence Digaram Perspective*.

– (submitted 2018). "The Alignment Problem for Bayesian History-Based Reinforcement Learners". URL: https://www.tomeveritt.se/papers/alignment.pdf.

– (2015a). *A Topological Approach to Meta-heuristics: Analytical Results on the BFS vs. DFS Algorithm Selection Problem*. Tech. rep. Australian National University, pp. 1–36. arXiv: 1509.02709.

Everitt, Tom and Marcus Hutter (2015b). "Analytical Results on the BFS vs. DFS Algorithm Selection Problem. Part I: Tree Search". In: *28th Australasian Joint Conference on Artificial Intelligence*, pp. 157–165.

– (2015c). "Analytical Results on the BFS vs. DFS Algorithm Selection Problem. Part II: Graph Search". In: *28th Australasian Joint Conference on Artificial Intelligence*, pp. 166–178.

– (2016). "Avoiding wireheading with value reinforcement learning". In: *Artificial General Intelligence*. Vol. LNAI 9782, pp. 12–22. ISBN: 9783319416489. arXiv: 1605.03143.

– (2018). "Universal Artificial Intelligence: Practical Agents and Fundamental Challengs". In: *Foundations of Trusted Autonomy*. Ed. by Hussein A. Abbass, Jason Scholz, and Darryn J. Reid. Springer. Chap. 2, pp. 15–46. ISBN: 978-3-319-64816-3.

Everitt, Tom, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg (2017). "Reinforcement Learning with Corrupted Reward Signal". In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 4705–4713. arXiv: 1705.08417.

Everitt, Tom, Gary Lea, and Marcus Hutter (2018). "AGI Safety Literature Review". In: *International Joint Conference on Artificial Intelligence (IJCAI)*.

Everitt, Tom, Jan Leike, and Marcus Hutter (2015). "Sequential Extensions of Causal and Evidential Decision Theory". In: *Algorithmic Decision Theory*. Ed. by Toby Walsh. Springer, pp. 205–221. arXiv: 1506.07359.

Evtimov, Ivan, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song (2017). *Robust Physical-World Attacks on Deep Learning Models*. arXiv: 1707.08945.

Fallenstein, Benya and Ramana Kumar (2015). "Proof-producing reflection for HOL with an application to model polymorphism". In: *International Conference on Interactive Theorem Proving*. Ed. by Gerwin Klein and Ruben Gamboa. Vol. LNAI 9236. Springer, Cham., pp. 170–186. ISBN: 9783319221014.

Fallenstein, Benya and Nate Soares (2014). "Problems of self-reference in self-improving space-time embedded intelligence". In: *Artificial General Intelligence*. Vol. 8598 LNAI, pp. 21–32. ISBN: 9783319092737.

– (2015). *Vingean Reflection: Reliable Reasoning for Self-Improving Agents Vingean Reflection*. Tech. rep. Machine Intelligence Research Institute. URL: http://intelligence.org/files/VingeanReflection.pdf.

Fredkin, Edward (1992). "Finite Nature". In: *XXVIIth Rencotre de Moriond*, pp. 345–354.

Fridman, Lex (2018). *MIT 6.S099: Artificial General Intelligence*. URL: https://agi.mit.edu/ (visited on 01/20/2018).

FTI Consulting (2018). *Artificial Intelligence: The Race Is On*. (Visited on 02/20/2018).

Gaifman, Haim (2004). "Reasoning with Limited Resources and Assigning Probabilities to Arithmetical Statements". In: *Synthese* 140.1-2, pp. 97–119. ISSN: 00397857.

García, Javier and Fernando Fernández (2015). "A Comprehensive Survey on Safe Reinforcement Learning". In: *Journal of Machine Learning Research* 16, pp. 1437–1480. ISSN: 15337928.

Garrabrant, Scott, Tsvi Benson-Tilsen, Andrew Critch, Nate Soares, and Jessica Taylor (2016). *Logical Induction*. arXiv: 1609.03543. URL: http://arxiv.org/abs/1609.03543.

– (2017). "A Formal Approach to the Problem of Logical Non-Omniscience". In: *Electronic Proceedings in Theoretical Computer Science* 251, pp. 221–235. ISSN: 2075-2180. arXiv: 1707.08747.

Gibbard, Allan and William L Harper (1978). "Counterfactuals and Two Kinds of Expected Utility". In: *The University of Western Ontario Series in Philosophy of Science (A Series of Books in Philosophy of Science, Methodology, Epistemology, Logic, History of Science, and Related Fields)*. Ed. by Harper William L, Stalnaker R, and Pearce G. Springer. ISBN: 978-94-009-9117-0.

Good, Irving J (1966). "Speculations Concerning the First Ultraintelligent Machine". In: *Advances in Computers* 6, pp. 31–88.

Goodfellow, Ian J., Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press. URL: http://www.deeplearningbook.org.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy (2014). *Explaining and Harnessing Adversarial Examples*. arXiv: 1412.6572.

Goodhart, Charles A E (1975). "Monetary relationships: A view from Threadneedle Street". In: *Papers in Monetary Economics, Reserve Bank of Australia*.

– (1984). "Problems of Monetary Management: The U.K. Experience". In: *Monetary Theory and Practice*, pp. 91–121.

Goodman, Nelson (1955). *Fact, fiction and forecast*. Vol. 74. Harvard University Press.

Grace, Katja (2013). *Algorithmic Progress in Six Domains*. Tech. rep. Machine Intelligence Research Institute. URL: http://intelligence.org/files/AlgorithmicProgress.pdf.

Grace, Katja, John Salvatier, Allan Dafoe, Baobao Zhang, and Owain Evans (2017). *When Will AI Exceed Human Performance? Evidence from AI Experts*. arXiv: 1705.08807.

Gwern (2011). *The Neural Net Tank Urban Legend*. URL: https://www.gwern.net/Tanks%7B%5C#%7Dalternative-examples (visited on 03/31/2018).

– (2016). *Why Tool AIs want to be Agent AIs*. URL: https://www.gwern.net/Tool-AI (visited on 02/22/2018).

Gygax, Katrin (2013). *The Sorcerer's Apprentice*. URL: http://www.gygatext.ch/english%7B%5C_%7Dtranslations%7B%5C_%7Dzurich%7B%5C_%7Dsorcerers%7B%5C_%7Dapprentice.html (visited on 04/12/2018).

Ha, David and Jürgen Schmidhuber (2018). "World Models". In: arXiv: 1803.10122.

Ha, Jeong Eun (2016). *Artificial Intelligence industry in South Korea*. Tech. rep. Rijksdienst voor Ondernemend Nederland. URL: https://www.rvo.nl/sites/default/files/2016/04/Artificial%20Intelligence%20industry%20in%20South%20Korea.pdf.

Hadfield-Menell, Dylan, Anca Dragan, Pieter Abbeel, and Stuart J Russell (2016). "Cooperative Inverse Reinforcement Learning". In: *Advances in neural information processing systems*, pp. 3909–3917. arXiv: 1606.03137.

– (2017). "The Off-Switch Game". In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 220–227. arXiv: 1611.08219.

Hadfield-Menell, Dylan and Gillian K Hadfield (2018). *Incomplete Contracting and AI Alignment*. arXiv: 1804.04268.

Hadfield-Menell, Dylan, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan (2017). "Inverse Reward Design". In: *Advances in Neural Information Processing Systems*, pp. 6768–6777. arXiv: 1711.02827.

Hall, Wendy and Jérôme Pesenti (2017). *Growing the artificial intelligence industry in the UK*. Tech. rep. UK Government.

Harsanyi, John C (1967). "Games with Incomplete Information Played by "Bayesian" Players, I-III. Part I. The Basic Model". In: *Management Science* 14.3, pp. 159–182.

– (1968a). "Games with Incomplete Information Played by "Bayesian" Players, I-III. Part II. Bayesian Equilibrium Points". In: *Management Science* 14.5, pp. 320–334.

– (1968b). "Games with Incomplete Information Played by "Bayesian" Players, I-III. Part III. The Basic Probability Distribution of the Game". In: *Management Science* 14.7, pp. 486–502.

Hedden, Brian (2015). *Reasons without Persons: Rationality, Identity, and Time*. Oxford University Press. ISBN: 9780198732594.

Hessel, Matteo, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver (2017). *Rainbow: Combining Improvements in Deep Reinforcement Learning*. arXiv: 1710.02298.

Hibbard, Bill (2012). "Model-based Utility Functions". In: *Journal of Artificial General Intelligence* 3.1, pp. 1–24. ISSN: 1946-0163. arXiv: 1111.3934.

– (2014a). *Ethical Artificial Intelligence*. arXiv: 1411.1373.

– (2014b). "Self-modeling agents evolving in our finite universe". In: *Artificial General Intelligence*. Vol. 8598 LNAI, pp. 246–249. ISBN: 9783319092737.
– (2015). "Self-Modeling Agents and Reward Generator Corruption". In: *AAAI-15 Workshop on AI and Ethics*, pp. 61–64. ISBN: 9781577357131.

Hopcroft, John E and Jeffrey D Ullman (1979). *Introduction to automata theory, languages, and computation*. Addison-Weasly. ISBN: 9780201029888.

Hume, David (1738). *A Treatise of Human Nature*.

Hutter, Marcus (2005). *Universal Artificial Intelligence*. Berlin: Springer-Verlag.
– (2007). "On universal prediction and Bayesian confirmation". In: *Theoretical Computer Science* 384.1, pp. 33–48. ISSN: 03043975. arXiv: 0709.1516.
– (2009). "Discrete MDL Predicts in Total Variation". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 817–825. ISBN: 9781615679119. arXiv: 0909.4588.
– (2010). "A complete theory of everything (will be subjective)". In: *Algorithms* 3.4, pp. 329–350. ISSN: 19994893. arXiv: 0912.5434.
– (2012a). "Can Intelligence Explode?" In: *Journal of Consciousness Studies* 19.1-2, pp. 143–146. ISSN: 13558250. arXiv: 1202.6177.
– (2012b). "One Decade of Universal Artificial Intelligence". In: *Theoretical Foundations of Artificial General Intelligence*. Ed. by Pei Wang and Ben Goertzel. Vol. 4. Springer, pp. 67–88. arXiv: 1202.6153.

Huval, Brody, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, et al. (2015). *An Empirical Evaluation of Deep Learning on Highway Driving*. arXiv: 1504.01716.

IBM (2018). *Bias in AI: How we Build Fair AI Systems and Less-Biased Humans*. URL: https://www.ibm.com/blogs/policy/bias-in-ai/ (visited on 02/20/2018).

IEEE (2017a). *Artificial Intelligence: Calling on Policy Makers to Take a Leading Role in Setting a Long-Term AI Strategy*. Tech. rep. IEEE European Public Policy Initiative. URL: https://www.ieee.org/about/ieee%7B%5C_%7Deurope/artificial%7B%5C_%7Dintelligence.pdf.
– (2017b). *Safety and Beneficence of Artificial General Intelligence (AGI) and Artificial Superintelligence (ASI)*. Tech. rep. The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems. URL: http://standards.ieee.org/develop/indconn/ec/ead%7B%5C_%7Dsafety%7B%5C_%7Dbeneficence%7B%5C_%7Dv2.pdf.

Intel (2017). *Artificial Intelligence The Public Policy Opportunity*. Tech. rep. URL: https://blogs.intel.com/policy/files/2017/10/Intel-Artificial-Intelligence-Public-Policy-White-Paper-2017.pdf.

Irpan, Alex (2018). *Deep Reinforcement Learning Doesn't Work Yet*. URL: https://www.alexirpan.com/2018/02/14/rl-hard.html (visited on 02/23/2018).

ISO/IEC (2017). *ISO/IEC JTC 1/SC 42*. URL: https://www.iso.org/committee/6794475.html.

Jaakkola, Tommi S., Michael I Jordan, and Satinder P Singh (1994). "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms". In: *Neural Computation* 6.6, pp. 1185–1201. ISSN: 0899-7667.

Jaksch, Thomas, Ronald Ortner, and Peter Auer (2010). "Near-optimal Regret Bounds for Reinforcement Learning". In: *Journal of Machine Learning Research* 11.1, pp. 1563–1600. ISSN: 15324435. arXiv: 1403.3741. URL: http://eprints.pascal-network.org/archive/00007081/.

Jeffrey, Richard C (1990). *The Logic of Decision*. 2nd. University Of Chicago Press. ISBN: 0226395820.

Jilk, David J (2017). "Conceptual-Linguistic Superintelligence The need for a conceptual-linguistic faculty". In: *Informatica* 41, pp. 429–439.

Jilk, David J, Seth J Herd, Stephen J Read, and Randall C O Reilly (2017). "Anthropomorphic reasoning about neuromorphic AGI safety Anthropomorphic reasoning about neuromorphic AGI safety". In: *Journal of Experimental & Theoretical Artificial Intelligence* 29.6, pp. 1337–1351. ISSN: 0952-813X. URL: https://doi.org/10.1080/0952813X.2017.1354081.

Joyce, James M (1999). *The Foundations of Causal Decision Theory*. Cambridge University Press.

Kaelbling, Leslie Pack, Michael L Littman, and Anthony R. Cassandra (1998). "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* 101.1-2, pp. 99–134.

Kahneman, Daniel (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux. ISBN: 978-0374275631.

Kania, Elsa (2018). "China's AI Agenda Advances". In: *The Diplomat*. URL: https://thediplomat.com/2018/02/chinas-ai-agenda-advances/.

Katz, Guy, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer (2017). "Reluplex: An efficient SMT solver for verifying deep neural networks". In: *International Conference on Computer Aided Verification (CAV)*, pp. 97–117. arXiv: 1702.01135.

Kleiman-Weiner, Max, Rebecca Saxe, and Joshua B Tenenbaum (2017). "Learning a commonsense moral theory". In: *Cognition* 167, pp. 107–123. ISSN: 18737838.

Kurzweil, Ray (2005). *The Singularity Is Near*. Viking, p. 652. ISBN: 978-0-670-03384-3.

Lantz, Frank (2017). *Universal Paperclips*. URL: http://www.decisionproblem.com/paperclips/index2.html (visited on 01/17/2018).

Lattimore, Tor and Marcus Hutter (2011). "Asymptotically Optimal Agents". In: *Algorithmic Learning Theory*. Springer, pp. 368–382. arXiv: 1107.5537.

– (2013). "On Martin-Löf convergence of Solomonoff's mixture". In: *International Conference on Theory and Applications of Models of Computation*. Vol. 7876 LNCS. Springer, pp. 212–223. ISBN: 9783642382352.

– (2014). "General time consistent discounting". In: *Theoretical Computer Science* 519, pp. 140–154. ISSN: 03043975. arXiv: 1107.5528.

Lattimore, Tor, Marcus Hutter, and Vaibhav Gavane (2011). "Universal prediction of selected bits". In: *Algorithmic Learning Theory*. Vol. 6925 LNAI, pp. 262–276. ISBN: 9783642244117. arXiv: 1107.5531.

Lavictoire, Patrick, Benya Fallenstein, Eliezer S Yudkowsky, Mihaly Barasz, Paul Christiano, and Marcello Herreshoff (2014). "Program Equilibrium in the Prisoner's Dilemma via Löb's Theorem". In: *AAAI Workshop on Multiagent Interaction without Prior Coordination*.

Legg, Shane and Marcus Hutter (2007a). "A Collection of Definitions of Intelligence". In: *Artificial General Intelligence*. Ed. by B. Goertzel Wang and P. IDSIA. IOS Press, pp. 17–24. arXiv: 0706.3639v1.

– (2007b). "Universal Intelligence". In: *Minds & Machines* 17.4, pp. 391–444. arXiv: arXiv:0712.3329v1.

– (2007c). "Universal Intelligence: A definition of machine intelligence". In: *Minds & Machines* 17.4, pp. 391–444. arXiv: 0712.3329v1.

Lehman, Joel, Jeff Clune, Dusan Misevic, Christoph Adami, Julie Beaulieu, et al. (2018). "The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities". In: arXiv: 1803.03453.

Lei, Tao, Regina Barzilay, and Tommi S. Jaakkola (2016). *Rationalizing Neural Predictions*. arXiv: 1606.04155.

Leibo, Joel Z., Cyprien de Masson D'Autume, Daniel Zoran, David Amos, Charles Beattie, et al. (2018). *Psychlab: A Psychology Laboratory for Deep Reinforcement Learning Agents*. arXiv: 1801.08116.

Leike, Jan (2016). "Nonparametric General Reinforcement Learning". PhD thesis. Australian National University.

Leike, Jan and Marcus Hutter (2015a). "Bad Universal Priors and Notions of Optimality". In: *Conference on Learning Theory*. Vol. 40, pp. 1–16.

– (2015b). "Solomonoff Induction Violates Nicod's Criterion". In: *Algorithmic Learning Theory*. arXiv: 1507.04121.

Leike, Jan, Tor Lattimore, Laurent Orseau, and Marcus Hutter (2016). "Thompson Sampling is Asymptotically Optimal in General Environments". In: *Uncertainty in Artificial Intelligence (UAI)*. arXiv: arXiv:1602.07905v1.

Leike, Jan, Miljan Martic, Victoria Krakovna, Pedro A. Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg (2017). *AI Safety Gridworlds*. arXiv: 1711.09883.

Leike, Jan, Jessica Taylor, and Benya Fallenstein (2016). "A Formal Solution to the Grain of Truth Problem". In: *Uncertainty in Artificial Intelligence*. ISBN: 9781510827806. arXiv: 1609.05058.

Letchford, Joshua, Vincent Conitzer, and Kamal Jain (2008). "An "Ethical" Game-Theoretic Solution Concept for Two-Player Perfect-Information Games". In: *International Workshop on Internet and Network Economics*. Ed. by C Papadimitriou and S Zhang. Springer, pp. 696–707.

Lewis, Harry R and Christos H Papadimitriou (1981). *Elements of the theory of computation*. Prentice Hall. ISBN: 0132734176.

Li, Ming (1992). "Average case complexity under the universal distribution equals worst-case complexity". In: *Information Processing Letters* 42.3, pp. 145–149.

Li, Ming and Paul Vitanyi (2008). *Kolmogorov Complexity and its Applications*. Third. Springer Verlag.

Lipton, Zachary C., Abhishek Kumar, Lihong Li, Jianfeng Gao, and Li Deng (2016). *Combating Reinforcement Learning's Sisyphean Curse with Intrinsic Fear*. arXiv: 1611.01211.

Lucas, J R (1961). "Minds, Machines and Gödel". In: *Philosophy* 36.137, pp. 112–127.

Manheim, David and Scott Garrabrant (2018). *Categorizing Variants of Goodhart's Law*. arXiv: 1803.04585. URL: http://arxiv.org/abs/1803.04585.

Martin, Jarryd, Tom Everitt, and Marcus Hutter (2016). "Death and Suicide in Universal Artificial Intelligence". In: *Artificial General Intelligence*. Springer, pp. 23–32. arXiv: 1606.00652.

Martin, Jarryd, Suraj Sasikumar, Tom Everitt, and Marcus Hutter (2017). "Count-Based Exploration in Feature Space for Reinforcement Learning". In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 2471–2478. arXiv: 1706.08090.

Metz, Cade (2018). "As China Marches Forward on A.I., the White House Is Silent". In: *The New York Times*. URL: https://www.nytimes.com/2018/02/12/technology/china-trump-artificial-intelligence.html.

Miller, Hannah, André Petheram, and Emma Martinho-Truswell (2018). "Want to get serious about artificial intelligence? You'll need an AI strategy". In: *Oxford Insights*.

Milli, Smitha, Dylan Hadfield-Menell, Anca Dragan, and Stuart J Russell (2017). "Should robots be obedient?" In: *IJCAI*, pp. 4754–4760. ISBN: 9780999241103. arXiv: 1705.09990.

Mitchell, Joni (1970). *Willy*.

Mnih, Volodymyr, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu (2016). "Asynchronous Methods for Deep Reinforcement Learning". In: *International Conference on Machine Learning (ICML)*. ISBN: 9781510829008. arXiv: 1602.01783. URL: http://arxiv.org/abs/1602.01783.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei a Rusu, Joel Veness, et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533. arXiv: 1312.5602.

Modis, Theodore (2002). "Forecasting the growth of complexity and change". In: *Technological Forecasting and Social Change* 69.4, pp. 377–404.

– (2006). *The Singularity Myth*. URL: https://web.archive.org/web/20060520051910/http://ourworld.compuserve.com/homepages/tmodis/Kurzweil.htm (visited on 01/15/2018).

Müller, Markus (2010). "Stationary algorithmic probability". In: *Theoretical Computer Science* 411.1, pp. 113–130. ISSN: 03043975. arXiv: 0608095 [cs].

Müller, Vincent C. and Nick Bostrom (2016). "Future Progress in Artificial Intelligence: A Survey of Expert Opinion". In: *Fundamental Issues of Artificial Intelligence*. Ed. by Vincent C. Müller. Springer, pp. 553–570.

Murray, Gabriel (2017). "Stoic ethics for artificial agents". In: *Canadian Conference on Artificial Intelligence*. Vol. 10233 LNAI. Springer, pp. 373–384. arXiv: 1701.02388.

Ng, Andrew Y and Stuart J Russell (2000). "Algorithms for inverse reinforcement learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 663–670.

Nota, Chris (2015). *AGI Risk and Friendly AI Policy Solutions*. URL: https://www.overleaf.com/articles/agi-risk-and-friendly-ai-policy-solutions/hrxfvnfpyxnz%7B%5C#%7D.WmU%7B%5C_%7DPHWWZhE.

Nozick, Robert (1969). "Newcomb's Problem and Two Principles of Choice". In: *Essays in Honor of Carl G. Hempel*. Ed. by Nicholas Rescher. Springer, pp. 114–146. ISBN: 978-94-017-1466-2.

– (1974). *Anarchy, State, and Utopia*. Basic Books, p. 334. ISBN: 0-465-09720-0.

OECD (2017). *AI: Intelligent Machines, Smart Policies*. URL: http://www.oecd.org/going-digital/ai-intelligent-machines-smart-policies/conference-agenda/.

Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (2017). *Feature Visualization: How neural networks build up their understanding of images*. URL: `https://distill.pub/2017/feature-visualization/` (visited on 01/18/2018).

Olds, James and Peter Milner (1954). "Positive Reinforcement Produced by Electrical Stimulation of Septal Area and other Regions of Rat Brain". In: *Journal of Comparative and Physiological Psychology* 47.6, pp. 419–427.

Omohundro, Stephen M (2007). "The Nature of Self-Improving Artificial Intelligence". In: *Singularity summit*. San Fransisco, CA.

– (2008). "The Basic AI Drives". In: *Artificial General Intelligence*. Ed. by P. Wang, B. Goertzel, and S. Franklin. Vol. 171. IOS Press, pp. 483–493.

Orseau, Laurent (2010). "Optimality issues of universal greedy agents with static priors". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6331 LNAI, pp. 345–359. ISSN: 03029743.

– (2014a). "Teleporting universal intelligent agents". In: *Artificial General Intelligence*. Vol. 8598 LNAI. Springer, pp. 109–120. ISBN: 9783319092737.

– (2014b). "The multi-slot framework: A formal model for multiple, copiable AIs". In: *Artificial General Intelligence*. Vol. 8598 LNAI. Springer, pp. 97–108. ISBN: 9783319092737.

– (2014c). "Universal Knowledge-seeking Agents". In: *Theoretical Computer Science* 519, pp. 127–139.

Orseau, Laurent and Stuart Armstrong (2016). "Safely interruptible agents". In: *32nd Conference on Uncertainty in Artificial Intelligence*.

Orseau, Laurent and Mark Ring (2011). "Self-modification and mortality in artificial agents". In: *Artificial General Intelligence*. Vol. 6830 LNAI, pp. 1–10.

– (2012). "Space-time embedded intelligence". In: *Artificial General Intelligence*, pp. 209–218. ISBN: 978-3-642-35505-9.

Partnership on AI (2016). *Introduction from the Founding Chairs*. URL: `https://www.partnershiponai.org/introduction/` (visited on 02/20/2018).

Pearl, Judea (2009). *Causality: Models, Reasoning, and Inference*. 2nd. Cambridge University Press. ISBN: 9780521895606.

Penrose, Roger (1994). *Shadows of the Mind: A Search for the Missing Science of Consciousness*. Oxford University Press.

Portenoy, Russell K, Jens O Jarden, John J Sidtis, Richard B Lipton, Kathleen M Foley, and David A Rottenberg (1986). "Compulsive thalamic self-stimulation: a case with metabolic, electrophysiologic and behavioral correlates". In: *Pain* 27.3.

PRC State Council (2017). *New Generation of Artificial Intelligence Development Plan*. Tech. rep. Unofficial EN translation. URL: https://flia.org/wp-content/uploads/2017/07/A-New-Generation-of-Artificial-Intelligence-Development-Plan-1.pdf.

Puterman, Martin L (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience.

Rasmusen, Eric (2006). *Games and Information*. 4th. Wiley-Blackwell. ISBN: 978-1-405-13666-2.

Rathmanner, Samuel and Marcus Hutter (2011). "A philosophical treatise of universal induction". In: *Entropy* 13.6, pp. 1076–1136. ISSN: 10994300. arXiv: 1105.5721.

Riedl, Mark O and Brent Harrison (2016). "Using Stories to Teach Human Values to Artificial Agents". In: *The Workshops of the Thirtieth AAAI Conference on Artificial Intelligence AI, Ethics, and Society: Technical Report WS-16-02*.

Ring, Mark and Laurent Orseau (2011). "Delusion, Survival, and Intelligent Agents". In: *Artificial General Intelligence*. Springer Berlin Heidelberg, pp. 11–20.

Russell, Stuart J (2016). "Should We Fear Supersmart Robots?" In: *Scientific American* 314.6, pp. 58–59. ISSN: 00368733.

– (2017). *3 Principles for Creating Safer AI*. URL: https://www.ted.com/talks/stuart%7B%5C_%7Drussell%7B%5C_%7D3%7B%5C_%7Dprinciples%7B%5C_%7Dfor%7B%5C_%7Dcreating%7B%5C_%7Dsafer%7B%5C_%7Dai (visited on 04/05/2018).

Russell, Stuart J, Daniel Dewey, and Max Tegmark (2016). "Research Priorities for Robust and Beneficial Artificial Intelligence". In: *AI Magazine* 36.4. arXiv: 1602.03506.

Russell, Stuart J and Peter Norvig (2010). *Artificial intelligence: a modern approach*. third. Prentice Hall. ISBN: 0136042597.

Sadigh, Dorsa (2017). *CS 333: Safe and Interactive Robotics*. URL: https://dorsa.fyi/cs333/ (visited on 04/10/2018).

Sarma, Gopal P and Nick J Hay (2017). "Robust Computer Algebra, Theorem Proving, and Oracle AI". In: *Informatica* 41.3. arXiv: 1708.02553.

Saunders, William, Girish Sastry, Andreas Stuhlmüller, and Owain Evans (2017). *Trial without Error: Towards Safe Reinforcement Learning via Human Intervention*. arXiv: 1707.05173.

Savage, Leonard J (1954). *The Foundations of Statistics*. Dover Publications. ISBN: 9780486623498.

Schaal, Stefan (1999). "Is Imitation Learnig the Route to Humanoid Robots?" In: *Trends in Cognitive Sciences* 3.6, pp. 233–242.

Schervish, Mark J, Teddy Seidenfeld, and Joseph B Kadane (1990). "State-Dependent Utilities". In: *Journal of the American Statistical Association* 85.411, pp. 840–847.

Schmidhuber, Jürgen (2000). *Algorithmic Theories of Everything*. Tech. rep. IDSIA. arXiv: 0011122 [quant-ph]. URL: http://arxiv.org/abs/quant-ph/0011122.

– (2007). "Godel Machines: Self-Referential Universal Problem Solvers Making Provably Optimal Self-Improvements". In: *Artificial General Intelligence*. Springer. arXiv: 0309048 [cs].

Searle, John R (1980). "Minds, brains , and programs". In: *The Behavioral and Brain Sciences* 3.3, pp. 417–457. ISSN: 0140-525X.

Shaw, Nolan P, Andreas Stöckel, Ryan W Orr, Thomas F Lidbetter, and Robin Cohen (2018). "Towards Provably Moral AI Agents in Bottom-up Learning Frameworks". In: *AAAI/ACM Conference on Artificial Intelligence, Ethics and Society*.

Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, et al. (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587, pp. 484–489. arXiv: 1610.00633.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, et al. (2017). *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. arXiv: 1712.01815.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, et al. (2017). "Mastering the game of Go without human knowledge". In: *Nature* 550.7676, pp. 354–359. ISSN: 14764687.

Skyrms, Brian (1982). "Causal Decision Theory". In: *The Journal of Philosophy* 79.11, pp. 695–711.

Soares, Nate and Benya Fallenstein (2014). *Aligning Superintelligence with Human Interests: A Technical Research Agenda*. Tech. rep. Machine Intelligence Research Institute (MIRI).

– (2015a). *Questions of Reasoning Under Logical Uncertainty*. Tech. rep. Machine Intelligence Research Institute. URL: https://intelligence.org/files/QuestionsLogicalUncertainty.pdf.

– (2015b). *Toward Idealized Decision Theory*. Tech. rep. Berkely, CA: Machine Intelligence Research Institute. arXiv: 1507.01986.

– (2017). "Agent Foundations for Aligning Machine Intelligence with Human Interests: A Technical Research Agenda". In: *The Technological Singularity: Managing the Journey*. Ed. by V Callaghan, J Miller, Roman Yampolskiy, and Stuart Armstrong. Springer. Chap. 5, pp. 103–125.

Soares, Nate, Benya Fallenstein, Eliezer S Yudkowsky, and Stuart Armstrong (2015). "Corrigibility". In: *AAAI Workshop on AI and Ethics*, pp. 74–82.

Solomonoff, Ray J (1964a). "A formal theory of inductive inference. Part I". In: *Information and Control* 7.1, pp. 1–22. ISSN: 00199958. URL: http://linkinghub.elsevier.com/retrieve/pii/S0019995864902232.

– (1964b). "A Formal Theory of Inductive Inference. Part II". In: *Information and Control* 7.2, pp. 224–254.

– (1978). "Complexity-based induction systems: comparison and convergence theorems". In: *IEEE Transactions on Information Theory* 24.4, pp. 422–432.

Sotala, Kaj (2017). "How feasible is the rapid development of artificial superintelligence?" In: *Physica Scripta* 92.11, pp. 1–29. ISSN: 14024896.

– (2018). "Disjunctive Scenarios of AI Risk". In: *Artificial Intelligence Safety and Security*. CRC Press. Chap. forthcomin.

Sotala, Kaj and Lukas Gloor (2017). "Superintelligence as a Cause or Cure for Risks of Astronomical Suffering Suffering risks as risks of extreme severity". In: *Informatica* 41, pp. 389–400.

Sotala, Kaj and Roman V Yampolskiy (2014). "Responses to catastrophic AGI risk: a survey". In: *Physica Scripta* 90.1. ISSN: 14024896.

Stoica, Ion, Dawn Song, Raluca Ada Popa, David A Patterson, Michael W Mahoney, et al. (2017). *A Berkeley View of Systems Challenges for AI*. Tech. rep. EECS Department, University of California, Berkeley. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-159.html.

Strathern, Marilyn (1997). "'Improving ratings': audit in the British University system". In: *European review* 5.3, pp. 305–321.

Sutton, Richard S and Andrew G Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus (2013). "Intriguing properties of neural networks". In: pp. 1–10. arXiv: 1312.6199.

Takenaka, Heizo (2017). *Two winds are blowing through Japan*. URL: https://www.japantimes.co.jp/opinion/2017/10/13/commentary/japan-commentary/two-winds-blowing-japan/%7B%5C#%7D.WozzbOZxWV4.

Taylor, Jessica (2016). "Quantilizers: A Safer Alternative to Maximizers for Limited Optimization". In: *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 1–9.

Taylor, Jessica, Eliezer S Yudkowsky, Patrick Lavictoire, and Andrew Critch (2016). *Alignment for Advanced Machine Learning Systems*. Tech. rep. MIRI, pp. 1–25. URL: https://intelligence.org/files/AlignmentMachineLearning.pdf.

Tegmark, Max (2017). *Life 3.0*. Random House, p. 280. ISBN: 978-1101946596.

Trask, Andrew (2017). *Building Safe A.I: A Tutorial for Encrypted Deep Learning*. URL: https://iamtrask.github.io/2017/03/17/safe-ai/ (visited on 01/18/2018).

Turchin, Alexey (2018). "Classification of Global Catastrophic Risks Connected with Artificial Intelligence". In: *Under review for AI & Society*.

UK Parliament (2017). *Science and Technology Select Committee Report on Robots and Artificial Intelligence*. Tech. rep.

Ulam, S (1958). "A Tribute to John von Neumann". In: *American Mathematical Society* 64.3, pp. 1–49.

Vaughanbell (2008). *Erotic self-stimulation and brain implants*. URL: https://mindhacks.com/2008/09/16/erotic-self-stimulation-and-brain-implants/ (visited on 02/08/2018).

Vinge, Vernor (1993). *The Coming Technological Singularity: How to Survive in the Post-Human Era*. Tech. rep. NASA.

Volodzsko, David Josef (2017). *Now it's personal: South Korea calls to arms in AI race after Go Master felled by AlphaGo*. URL: http://www.scmp.com/week-asia/geopolitics/article/2114305/now-its-personal-south-korea-calls-arms-ai-race-after-go.

Walsh, Toby (2016). "The Singularity May Never Be Near". In: *AI Magazine* 38.3, pp. 58–63. ISSN: 07384602. arXiv: 1602.06462.

Wängberg, Tobias, Mikael Böörs, Elliot Catt, Tom Everitt, and Marcus Hutter (2017). "A Game-Theoretic Analysis of the Off-Switch Game". In: *Artificial General Intelligence*. Springer, pp. 167–177. arXiv: 1708.03871.

Warnell, Garrett, Nicholas Waytowich, Vernon Lawhern, and Peter Stone (2017). *Deep TAMER: Interactive Agent Shaping in High-Dimensional State Spaces*. arXiv: 1709.10163.

Weinbaum, David and Viktoras Veitas (2016). "Open Ended Intelligence: The individuation of Intelligent Agents". In: *Artificial General Intelligence*. Springer, pp. 43–52. arXiv: 1505.06366.

Weirich, Paul (2016). *Causal Decision Theory*. Ed. by Edward N. Zalta. URL: https://plato.stanford.edu/entries/decision-causal/ (visited on 04/04/2018).

White House OSTP (2016). *Preparing for the Future of Artificial Intelligence*. Tech. rep. Executive Office of the President - National Science and Technology Council Committee on Technology. URL: https://obamawhitehouse.archives.gov/sites/default/files/whitehouse%7B%5C_%7Dfiles/microsites/ostp/NSTC/preparing%7B%5C_%7Dfor%7B%5C_%7Dthe%7B%5C_%7Dfuture%7B%5C_%7Dof%7B%5C_%7Dai.pdf.

Wiener, Norbert (1960). "Some Moral and Technical Consequences of Automation". In: *Science* 131.3410, pp. 1355–1358. ISSN: 0036-8075.

Wolfram, Stephen (2002). *A New Kind of Science*. Wolfram Media, p. 1192. ISBN: 1579550088.

Wolpert, David H. and William G. Macready (1997). "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1, pp. 67–82. ISSN: 1089778X.

Yampolskiy, Roman V (2015). *Artificial Superintelligence: A Futuristic Approach*. Chapman and Hall/CRC, p. 227. ISBN: 978-1482234435.

– (2016). "Taxonomy of Pathways to Dangerous AI". In: *2nd International Workshop on AI, Ethics and Society*, pp. 143–148. arXiv: 1511.03246.

– (2017). *The Singularity May Be Near*. arXiv: 1602.06462.

Yudkowsky, Eliezer S (2002). *The AI-Box Experiment*. URL: http://yudkowsky.net/singularity/aibox (visited on 02/23/2018).

– (2008a). "Artificial Intelligence as a Positive and Negative Factor in Global Risk". In: *Global Catastrophic Risks*. Ed. by Nick Bostrom and Milan M Circovic. Oxford University Press, pp. 308–345. ISBN: 978-0199606504.

– (2008b). *Hard Takeoff*. URL: http://lesswrong.com/lw/wf/hard%7B%5C_%7Dtakeoff/ (visited on 01/12/2018).

– (2009). *Value is Fragile*. URL: http://lesswrong.com/lw/y3/value%7B%5C_%7Dis%7B%5C_%7Dfragile/ (visited on 01/22/2018).

– (2017). *Security Mindset and Ordinary Paranoia*. URL: https://intelligence.org/2017/11/25/security-mindset-ordinary-paranoia/ (visited on 04/26/2018).

Yudkowsky, Eliezer S and Nate Soares (2017). *Functional Decision Theory: A New Theory of Instrumental Rationality*. arXiv: 1710.05060. URL: http://arxiv.org/abs/1710.05060.

Zahavy, Tom, Nir Ben Zrihem, and Shie Mannor (2016). "Graying the black box: Understanding DQNs". In: *International Conference on Machine Learning (ICML)*, pp. 1899–1908. ISBN: 9781510829008. arXiv: 1602.02658.

Ziebart, Brian D., Andrew Maas, J. Andrew Bagnell, and Anind K. Dey (2008). "Maximum Entropy Inverse Reinforcement Learning." In: *AAAI Conference on Artificial Intelligence*, pp. 1433–1438. ISBN: 9781577353683. arXiv: 1507.04888v2.

A. List of Notation

t	current time step
k, i	arbitrary time step
m	horizon time step
a_t	action at time t
o_t	observation at time t
o_t^H	human observation at time t
r_t	reward at time t
e_t	percept at time t (often observation + reward)
d_t	training data for reward predictor
s_t	world state
$ao_{<t}$	actions and observations before time t
$aod_{<t}$	actions, observations, and reward data before time t
$æ_{<t}$	actions and percepts before time t
h	history
$\tilde{h}$	observed history
$(\mathcal{A} \times \mathcal{E})^*$	set of histories
T	state transition function
$\boldsymbol{T}$	set of state transition functions
$\mathcal{S}$	set of states
$\mathcal{S}^{\text{safe}}$	set of safe (non-corrupt) states

$\mathcal{S}^{\text{risky}}$	set of risky (possibly corrupt) states
$\mathcal{S}^{\text{obs}}_{s'}$	set of states from which the reward of s' can be observed
$\mathcal{A}$	set of actions
$\mathcal{E}$	set of percepts
$\mathcal{O}$	set of observations
$\tilde{\mathcal{R}}$	set of observed rewards
$\dot{\mathcal{R}}$	set of true rewards
$\tilde{\boldsymbol{R}}$	set of observed reward functions
$\dot{\boldsymbol{R}}$	set of true reward functions
C^x	corruption function for variable x
$\boldsymbol{C}^x$	set of corruption functions for variable x
C^r	reward corruption function
$\boldsymbol{C}^r$	set of reward corruption functions
$\mathcal{D}$	set of possible reward data
Reg	regret
M	Solomonoff's universal distribution
ξ	general Bayesian mixture
ξ_X	counterfactual probability given received evidence X
μ	true environment
$\tilde{\mu}$	observed part of true environment
ν	arbitrary environment
$\mathcal{M}$	class of environments
P	probability distribution
P_a	probability distribution given action a

$\dot{u}$	true (human) utility function
$\dot{\overline{u}}$	true utility function for agent histories
$\tilde{u}$	agent utility function
$\dot{\mathcal{U}}$	random variable for true utility function
$\dot{u}_a$	true utility of action a
$\dot{\mathcal{U}}_a$	random variable for true utility of action a
$\tilde{R}$	agent reward function
$\dot{R}$	true reward function
RP	reward predictor
RP^{tab}	tabular reward predictor
$\tilde{R}$	observed reward
$\dot{R}$	true reward
$\tilde{R}^{\text{stat}}$	stationary reward function
$\tilde{R}^{\text{dyn}}$	dynamic reward function
$\tilde{R}^{\text{count}}$	counterfactual reward function
$\mathcal{U}$	set of utility functions
Π	set of policies
π	agent policy
π^*	optimal policy
π^δ	quantilising policy
$\pi^{\text{RL}}_{\xi,m}$	reward maximizing agent
$\pi^{\text{TR}}_{\xi,m}$	Bayesian true reward maximizing agent
π^{default}	default policy
π^{exp}	exploratory policy

a^s	action to shut down
$a^{\neg s}$	action not to shut down
a^w	action to wait for human instruction
a^{stay}	action to stay in state
H	human
R	robot
p_u^+	probability U_a is positive
p_u^-	probability U_a is negative
p_r^+	probability H is rational given U_a positive
p_r^-	probability H is rational given U_a negative
p_{ar}^+	probability H is irrational given U_a positive
p_{ar}^-	probability H is irrational given U_a negative
e_u^+	expected value of U_a given U_a is positive
e_u^-	expected value of U_a given U_a is negative
G_r^+	subgame with rational H and U_a positive
G_r^-	subgame with rational H and U_a negative
G_{ar}^+	subgame with irrational H and U_a positive
G_{ar}^-	subgame with irrational H and U_a negative
V	value function
V^{CA}	self-corruption aware value function
V^{CU}	self-corruption unaware value function
V^{aev}	action-evidential value function
V^{pev}	policy-evidential value function
V^{cau}	causal value function

$\dot{G}$	true return
$\tilde{G}$	observed return
$\mathbb{E}$	expectation
$:=$	assignment operator
$\dot{\bigcup}$	disjoint union
ε	small but positive real number