On Problem Specification and Self-Referential Claims

by

Michael D Dennis

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Stuart Russell, Chair
Associate Professor Anca Dragan
Associate Professor Sergey Levine
Professor Michael Littman

Fall 2022

On Problem Specification and Self-Referential Claims

Abstract

On Problem Specification and Self-Referential Claims

by

Michael D Dennis

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Stuart Russell, Chair

Before applying an AI system to any real-world problem, it is first necessary to explicitly specify an objective for the system to solve. Whether this is a loss function over a dataset, an MDP to plan in, or a simulator in which to train an RL agent, we must eventually write down a specification that reduces our vague real-world problem to an explicit computational problem. In this thesis, I view writing an explicit problem specification as an act of communication. In this way, a problem specification is more effective if it better achieves the ends of the AI designer in communicating the true problem specification.

We begin by formalizing the problem faced by the designer, which we call the Specification Design Problem. In this problem, the designer must choose between a set of available problem specifications to be solved and aims to select one whose solution performs well on the designer's true unspecified problem. The designer's goal in the specification design problem is to minimize the specification error – the utility the designer has lost by simply writing down the wrong problem. We show that specification error decomposes into underspecification error, caused by leaving out helpful information, and misspecification error, caused by specifying incorrect information. Furthermore, we can analyze each of these errors in turn. We can characterize underspecification error through a new measure of the value of information. Alternatively, we can understand misspecification error through a geometric analysis of the crucial trade-offs of a particular domain. Taken together, these tools allow us to provide recommendations to the AI designer, which we demonstrate by determining the conditions under which the AI designer should stop specifying more information.

In the second part, we use this perspective to study a class of problem specifications often used in practice to mitigate misspecification errors. Such specifications, including maximizing worst-case utility or minimizing worst-case regret, are often used to avoid specifying complex features of the true problem and produce a system robust to that feature. These approaches contrast the Bayesian framework, which requires fixing some distribution for that feature. However, despite their practical use, it has yet to be shown that such approaches

are strictly necessary. While they improve the performance of current systems, one may hope we could eventually replace them with an appropriate model of uncertainty within the Bayesian framework. In this chapter, we provide an example where a specification resembling maximizing worst-case utility is practically unavoidable, as any explicit specification within the standard Bayesian framework requires specifying the task in excessive and impractical detail. This example shows that, counterintuitively, the optimal solution for a Bayesian AI designer in a specification design problem may be to create a non-Bayesian agent.

To match the needs of the AI designer in such specification design problems, I provide an extension to the Bayesian framework to include these helpful problem specifications. The core difficulty of this extension is that these specifications make what I call self-referential claims – assertions about how the policy will perform in the world. For instance, maximizing worst-case utility can be interpreted as the claim that "the world is one of the worst for your policy" and minimizing worst-case regret can be interpreted as the claim "the policy is likely to incur high loss". Thus self-referential claims allow us to study these specifications as a class rather than individually. In addition, it enables us to create specifications that utilize other natural self-referential claims like "you are unlikely to be able to fix the engine", "you are likely to be successful", or "you are unlikely to make money on the stock market". Such claims are natural in everyday conversation, partly because they efficiently communicate aspects of the problem at hand, describing essential elements of the engine or market conditions without needing to describe in intricate detail how those systems work. As such, their use is critical to the efficient solution of a wide range of specification design problems.

While there are numerous subtleties in integrating self-referential claims into the Bayesian framework, they can be carefully navigated by developing two formalisms analogous to causal and evidential decision theory. We characterize their edge cases and provide general conditions under which these formalisms produce equivalent results. Thus we can use self-referential claims to build the natural, efficient, and robust specifications we need as AI designers while maintaining philosophical and mathematical rigor.

Finally, I discuss how the problem specifications discussed in this theory can be effectively solved by scalable RL systems through automatically designing training environments. We formalize this approach as Unsupervised Environment Design (UED) and show that it can find policies consistent with the theory of self-referential claims. Moreover, we propose an algorithm called Protagonist Antagonist Induced Regret Environment Design (PAIRED), which finds minimax regret strategies as the Nash equilibrium of a 3-agent system. Finally, we show that this approach has benefits for promoting the transfer of the resulting policy and results in a natural automatic curriculum of increasing complexity.

Overall, through analyzing the specification design problem, understanding the effectiveness of self-referential claims in problem specifications, and allowing for their scalable implementation through unsupervised environment design, this thesis lays the necessary groundwork for a study of problem specification, well-grounded theoretically, empirically, and practically.

To my friends and family who supported me along the way.

Contents

List of Figures

List of Tables

Acknowledgments

Throughout my research journey, many people were critical to my success. They challenged my perspective, helped me grow as a researcher, and supported me as a person. In all of these ways, I thank Ljubomir Perkovic, without whom I would not be the person I am today and likely would never have been introduced to research. I especially appreciate his teaching me to be persistent in the face of complex problems and for demonstrating how to approach research and life with a deep, unrelenting kindness which I aspire to take into every collaboration and every day. His advice and support on issues far removed from our collaboration will always be remembered.

I am also grateful to Jonathan Shewchuk for supporting me throughout my research journey. I am thankful for the help growing as a research communicator and support in my development as a researcher, even as my interests shifted.

Of course, I am immensely grateful for the continual support of my research advisor Stuart Russell. I appreciate his supporting my academic freedom, allowing me to pull threads and fix holes in my understanding. Despite these threads sometimes seeming excessively pedantic and outside the realm of usual AI research, he appreciated these efforts not despite these quirks but because of them. In prioritizing fundamental understanding over publications, he helped me learn to trust my instincts, take my confusion seriously, and find intrinsic value in my understanding.

I want to thank the members of the vibrant research community, each of whom cares deeply about doing good in the world and who have challenged me to build my understanding of the field and develop my perspective: Adam Gleave, Rohin Shah, Daniel Filan, Dylan Hadfield-Menell, Tom Gilbert, Micah Carroll, Olivia Watkins, Sam Toyer, Scott Emmons, Lawrence Chan, Cassidy Laidlaw, Niklas Lauffer, Li, Rachel Freedman, Anand Siththaranjan, George Matheos, Jonathan Stray.

I am indebted to my colleague and collaborators who have helped me develop my direction and vision through their thoughtful conversations and advice: Natasha Jaques, Eugene Vinitsky, Minqi Jiang, Jack Parker-Holder, Cody Wild, Jakob Foerster, Edward Grefenstette, Tim Rocktäschel, Andrew Critch, Sergey Levine, Tom Lenaerts, Cathy Wu.

I am equally indebted to those I have advised, who have helped me grow as a mentor and researcher: Yawen Duan, Charlotte Roman, Joe Benton, Neel Alex, Rafael Albert, Johannes Treutlein, Michelle Li, Mikayel Samvelyan, and Ishita Mediratta.

Grad school would not have been as fun without any one of these amazing people: Stephanie Wang, Tyler Westenbroek, Michael Chang, Rohan Padhye, Gautam Gunjala, Stan Smith, Kevin Laeufer, Kristina Monakhova, Ben Brock, Alex Reinking, David Gaddy, Alain Anton. For backpacking trips, parties, carpentry projects, heart-to-heart chats, walks, and coffees, I am deeply grateful to every one of you.

Finally, I thank Jeannie Wilkening for always being there, putting things in perspective, and helping me keep my head on straight.

Chapter 1

Introduction

Any application of an AI system to solve a real-world problem must begin with a formal and explicit specification of the problem to be solved. Whether this is finding an optimal policy for an explicitly written MDP, parameters of a CNN locally minimizing loss on MNIST, or the solution to a satisfiability problem, every method we have in AI requires such an explicit problem specification. To be precise, we will use the term **explicit problem specification** to refer to the formal description of the real-world problem as an equation to be solved. Such a specification includes any necessary empirical data, such that any vagueness or empirical uncertainty has been managed explicitly. Thus, given an explicit problem specification, the only remaining challenge is computational. We call the *problem specification paradigm* – the two-step process of translating the true real-world problem specification into an explicit problem specification and then solving this specification to find an acceptable solution for the true problem. This paradigm is central to our ability to leverage our immense wealth of computation and algorithmic knowledge to solve pressing real-world problems.

The utility of an explicit problem specification comes from its ability to decompose the task of solving a problem into two steps, **specification** and **computation**. As computer scientists, we focus primarily on effectively performing computation, taking the specification for granted. This is productive as long as solving important problems faces a **computational bottleneck** – where it is easy to explicitly specify important problems which were amenable to computational progress. However, in the past decade, we have been met with an astounding **decreasing cost to computation**. Fueled by the exponential growth in computational efficiency [64, 75], algorithmic improvements [27], software optimization [68], and data availability [76], the past decade has been so rife with unanticipated breakthroughs in image classification [36], natural language processing [11], generative models [55], Go [70, 71], Poker [10, 47], that to point it out borders on clechè. It seems our field has become so effective at what we do that any useful explicit problem specification we cannot currently solve will be met with innovative computational solutions in short order.

Simultaneously, as our systems are applied to increasingly ambitious real-world applications, we find ourselves faced with an **increasing cost of specification**. We see systems repeatedly doing what the designer specified instead of doing what they intended. We find that algorithms learn biases from their datasets in recidivism prediction, job application screening, image recognition, or language modeling, we find that algorithms pick up on spurious correlations in applications like medical imaging, we find that they leak private information, we find that we cannot adequately detect unsafe or unanticipated situations in self-driving cars or prepare for the long-tail of difficult to anticipate failure modes. Correcting these problems requires specifying vague ideas like fairness, justice, privacy, or safety, which resist formalization. Even in the lab, the bottleneck in many active areas of AI research is the difficulty of specifying some core concept correctly, though it is not often referred to in those terms. To see this, imagine the progress we would make if we could specify the "prior over MDPs" in exploration, "the ideal next learning task" in curriculum learning, "the true reward function" in AI Alignment, "the correct measure of human understanding" in ML interpretability "the optimal equilibrium selection procedure" in Multi-Agent RL, or "the correct human model" in HRI. The fact that so many of the fundamental bottlenecks to progress in our field revolve around the difficulty of specifying some part of the problem should be enough to justify a concerted investigation into the challenge of problem specification. Taken together, it becomes clear that using computation to solve real-world problems often faces a **specification bottleneck** – where much of the challenge in using computation to solve a problem will be in explicitly specifying the problem itself.

The way that we should design systems under a specification bottleneck looks radically different than under a computational bottleneck. If we can solve any problem we specify, it seems unlikely we would decide to specify problems in terms of reward functions, MDPs, priors, or linear programs. Though each of these specifications admits tractable algorithms, they are quite divorced from how problems are described in everyday conversation. Thus, they offload the monumental effort of specifying the real-world task to the AI practitioner. These specifications were designed under an **illusory separation of concerns** which imagines that the language of problems we decide to build algorithms to solve does not impact the ease with which we can translate the real-world problems we care about into that language. However, in a world facing a specification bottleneck, we must reject this separation of concerns – rather than design our specifications to admit effective algorithms, we must design our algorithms around useful specifications. As an analogy, consider the case of designing fighter jets. Under optimal pilot controls, the theoretical limit to the speed and maneuverability of a fighter jet is much higher than what is achieved in today's systems. However, if we built the theoretically optimal jet, it would be impossible for any human to control due to the limited reaction time, being unable to correct instabilities until it was too late – thus though it is a better system, in theory, no human can specify the instructions which would cause those benefits to manifest. Though it may be tenable for some jets to label crashes as "pilot error," it becomes untenable to do this at some point before the plane cannot be flown by any human pilot. Modern fighter jets have thus been designed around what the human pilot can accurately specify. From this, we should take that when faced with a specification bottleneck, any effective system must be designed around useful specifications. This requires accounting for the practical difficulty inherent in specifying the problems we care about.

This leads to this thesis's central question: How can we build useful specifications? The

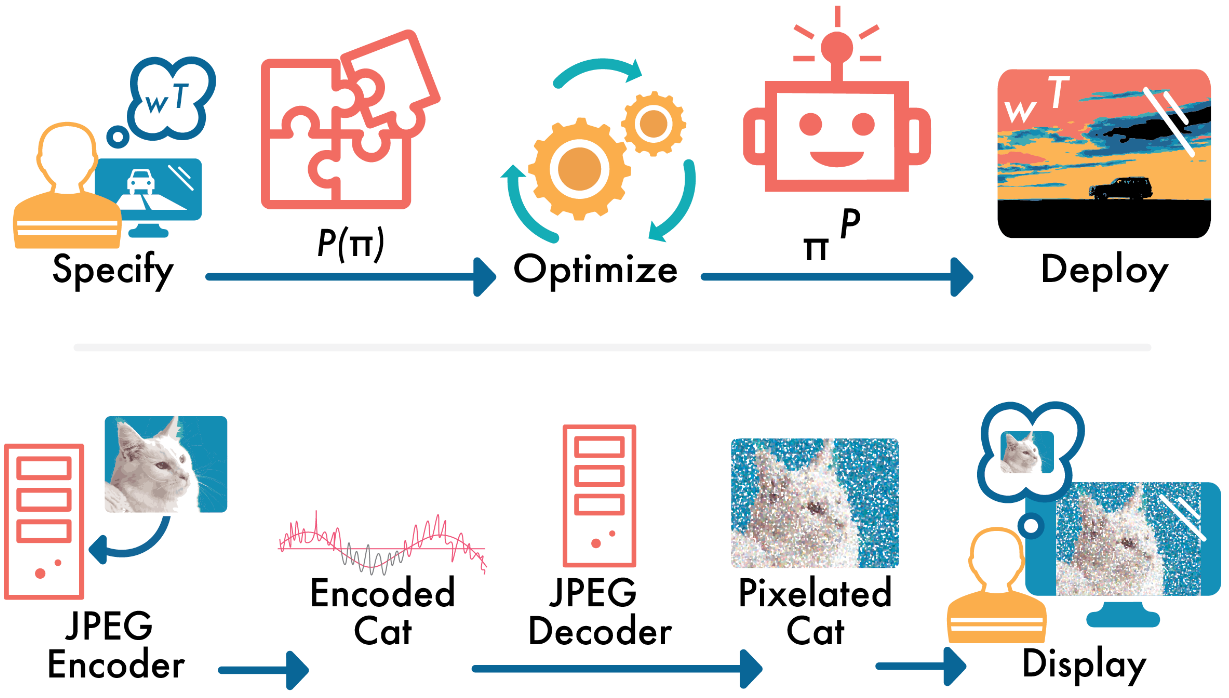


Figure 1.1: A depiction of a specification design problem, shown on top, where the designer specifies an explicit problem specification, which is provided to an optimization algorithm to result in a policy deployed into the real world – following the problem specification paradigm. This process is analogous to lossy compression algorithms, shown below, such as JPEG compression, in which the true image is compressed, and decompressed, removing information about the true image but preserving the perception of the image seen by the human observer.

difficulty of this question resides in **the central tension of problem specification** between the famous sayings of Box and Goodhart. Box coined a famous adage, “all models are wrong ...”, which applies equally well to problem specifications as they are models of the real-world problem [9]. However, if the specification we solve is wrong, we must contend with another famous adage by Goodhart, “When a measure becomes a target, it ceases to be a good measure,” which amounts to a claim that solving any wrong problem specification will lead to a poor solution to the designer’s intended problem [41]. Taken together, Goodhart and Box imply that it is impossible to build beneficial AI systems for real-world problems since every explicit specification would be wrong and thus lead to poor performance. Therefore, to make progress, we are forced to reinterpret Goodhart’s Law not as a law of nature but as a ”tendency” in the same way we interpret the folk law of gravity “What goes up, must come down”. In this sense, we must build our specifications to defy Goodhart’s law the same way bridges and skyscrapers can be said to defy gravity. In this way, we can build problem specifications that, though they are wrong, are still useful.

To build intuition about the nature of the problem, it is useful to understand an explicit problem specification as a message between the AI designer and the computer. In this thesis, we will model this situation formally as a *specification design problem*, depicted in Figure 1.1, and formally described in Chapter 2. Box's observation is analogous to the fact that explicit problem specifications constitute an act of lossy communication. Much like a JPEG encoder compresses an image into a file which a JPEG decoder then uses to make an image that appears similar to the original, an AI designer should compress a problem specification into an explicit problem specification such that the policy received from optimizing the explicit problem specification is approximately optimal on the true one.

Moreover, just as the JPEG encoder works by throwing out the unimportant details (the high-frequency information) and letting the JPEG decoder guess this information (by falling the low-frequency components), our explicit problem specifications must do the same, leaving out unimportant details about the problem allowing the computer to fill in the gaps. For JPEG compression, it is common knowledge between the encoder and decoder that the low-frequency information is unimportant for the human perception of the image. Achieving the same success in problem specification will require understanding which details of a problem can safely be ignored.

Viewing problem specification as an act of communication, we use Chapter 2 to formally define specification design problems and use them to study how the AI designer can create efficient and effective specifications. In Chapter 3, we use this perspective to analyze a specific class of problem specifications that include what we call "self-referential claims," often used to mitigate misspecification. These specifications, including maximizing worst-case utility or minimizing worst-case regret, allow the AI designer to avoid specifying complex features of the true problem by producing a system robust to that feature. We provide examples showing that self-referential claims can be critical to the AI designer, thus showing they are essential for effective performance in specific specification design problems. Finally, in Chapter 4, we describe how the problem specifications discussed in the other chapters can be optimized effectively using unsupervised environment design. In the rest of this introduction, we will provide an overview of the contributions of this thesis in more detail.

1.1 Efficient and Effective Specification via Specification Design Problems

In Chapter 2, we formalize the problem specification paradigm as *specification design problems*. These problems formally model the decision faced by the AI designer who must choose between a set of available explicit problem specifications. The chosen problem specification will then be optimized with the aim of performing well on the designer's true problem. This problem abstracts away the practical and computation issues of solving the explicit problem specification which the rest of the field focuses on, to instead focus on the problem of specification.

The usefulness of the specification design problem formalisms comes from the fact that it provides a definition of what it means to have a good problem specification – a good problem specification is one that is instrumentally useful for the designer. Thus, the central quantity the AI designer aims to minimize is the *specification error*, the amount of error the AI designer has incurred simply by writing down the wrong problem. In the remainder of the Chapter, we analyze the specification design problem by decomposing specification error into two terms, underspecification error, the error incurred by information that has been left out of the specification, and misspecification error, the error incurred by information that has been incorrectly specified.

Underspecification error, the error incurred by information that has been omitted from the specification, can be analyzed by constructing a measure of the value of information which we call specification value. The specification value is the difference between the value the system would have achieved by knowing some claim X, and the value that it would have achieved without knowing that claim.

We show that underspecification error can be rewritten in terms of the specification value of the omitted claims. Moreover, we show that the specification value of any set of claims behaves intuitively, in the same way as measuring area in a Ven diagram. Thus specification value offers an intuitive way of understanding what should be specified to most quickly reduced the underspecification error.

To analyze misspecification error, we offer an alternative approach. We notice that misspecification error is high when there exists an important trade-off with respect to the variable that was misspecified. When looking at the Pareto frontier, depicting how all available policies perform on the dimension being misspecified, high misspecification error corresponds to a Pareto frontier with low curvature. In fact, we can make this formal by modeling the Pareto curve geometrically, and thus can provide a tool to rigorously analyze misspecification error by characterizing this curvature.

Finally, we combine the analysis of underspecification error using specification value, and the analysis of misspecification error using the curvature of the Pareto frontier, to analyze the specification error of the designer and provide recommendations for how the designer ought to specify problems. As a demonstration, we analyze the problem of when the designer should stop specifying. Through analyzing a concrete case we show that, since the optimal designer would choose the most valuable information to specify first, the underspecification error decreases rapidly at the beginning before having reduced marginal decreases. Moreover, since the optimal designer would tend to stay away from variables that are easily misspecified, the specification error would begin small and have increased marginal increases, leading to higher misspecification error over time. As the specification error is the sum of the two, this leads to an optimal point in the middle where the marginal decrease in underspecification error is dominated by the marginal increase in misspecification error. This leads to a phenomenon similar to the bias-variance trade-off.

Overall, we show that the specification design problem formalism can lead to productive insights. By showing that specification error can be decomposed into underspecification and misspecification error, and providing communication and geometric tools to analyze each in turn, we provide a scaffolding to studying the decisions an AI designer ought to make in a given application. Moreover, we show that these tools can be applied to supply a concrete recommendation for AI designers, recommending stopping with an underspecified problem when the marginal decrease in underspecification is smaller than the marginal increase in misspecification. Thus our framework provides the groundwork necessary to rigorously analyze problem specifications and come to concrete recommendations for practitioners.

1.2 The Necessity of Self-Referential Claims for Efficient Specification

In Chapter 3 we focus on a class of problem specifications often used to mitigate misspecification errors. This class includes specifications like worst-case analysis and minimax regret, which fall outside the traditional Bayesian framework of AI. Such specifications tend to be useful in practice when handling parameter of the specification that are difficult to correctly specify, allowing the designer to instead choose to build a system that would be robust to any value of the parameter. However, despite their practical utility, it is unclear whether such deviations from the traditional Bayesian framework are ever strictly necessary. Intuitively, it may seem that one could choose instead to be appropriately uncertain over these unspecified parameters within the Bayesian model, and thus all problem specifications could be optimally and practically done within the Bayesian framework. Thus a key question in our understanding of problem specifications broadly is whether a good solution to a specification design problem ever requires deviations from the traditional Bayesian framework.

In this section, we provide an example showing that such deviations are sometimes necessary. That is, we present a problem that is easy to specify by asking for a policy that maximizes worst-case utility, but is difficult to specify by asking for a policy that maximizes expected value in the traditional Bayesian model. This shows, counterintuitively, that a traditional Bayesian AI designer in a specification design problem will often be best served by creating a non-Bayesian agent.

Intuitively a specification built around worst-case utility can be easier to specify because, specifying a Bayesian problem specification would require writing down a belief, and all simple beliefs that a designer can practically write down could be importantly wrong. If, at the same time, worst case performance may be approximately optimal for the particular problem in question, it may be easier to simply ask the system to optimize the worst case performance. For example, both of these conditions end up being true in Chess. Any simple model of the opponent is not going to play perfect chess and thus will produce an agent that does not need to play perfect chess. Simultaneously, the optimal policy against a worst-case chess player will work well against any chess player. Thus, for the natural ways of formulating chess as a specification design problem, the designer would prefer a worst-case specification over any simple Bayesian model.

To provide a specification language that can effectively serve the designer's needs in such situations, we use the rest of the chapter, to detail an extension to the Bayesian framework which allows for these sorts of specifications. Specifically, these are specifications that contain what we call self-referential claims. Self-referential claims are claims about a property of your policy in the world. For example, “you will not make that jump”, “you overestimate your abilities as a stand-up comic”, or “you are unlikely to make money on the stock market”. Such claims are, of course, critical to everyday life, as many plans would fail or opportunities would be missed if one did not have an accurate conception of their own abilities. They also allow for the expression of the previously discussed specifications, worst-case analysis is expressed by the self-referential claim “the world is one of the worst for your policy” and minimizing worst-case regret is expressed by the self-referential claim “the policy is likely to incur high loss”.

Their utility of self-referential claims comes from the fact that certain specification design problems are difficult without them. These claims often more efficiently communicate problems as it is easier to communicate directly about the capabilities or incapabilities of a system than it is to explain why those capabilities exist. For instance, it may be easier to tell your friend that they will be unlikely to make money on the stock market than it would be to explain the market forces that would make all of their specific investment schemes fail. This sheds more light on why specifications including self-referential claims would be necessary to effectively solve certain specification design problems. If the only problem with your AI system is it attempts a task, like trading on the stock market, for which it is ill-equipped, then being able to specify “you are unlikely to be successful with this task” is relatively easy to specify, but explaining why they are unlikely to be successful in direct terms could be arbitrarily complex.

However, though self-referential claims are critical to the success of any agent, and though traditional Bayesian agents will have beliefs about self-referential claims, such beliefs cannot be directly specified in the standard framework. That is, beliefs about self-referential claims are always derived from beliefs about the model and the proposed strategy.

For instance, a traditional Bayesian agent may have a belief that they will be unable to make money on the stock market, but only if that inability is derived from their beliefs about market conditions. If an agent mistakenly believed that they could make money on the stock market, we could not directly correct them. The only way to convince them that they would be to explain why their strategies would not work, by explaining market conditions. However, every time you discredited a strategy the agent could believe in another. You could tell them that the market is perfectly efficient, and thus there is no money to be made. However, this would be a lie, as some agents with more knowledge do make money consistently. Moreover, the only relevant detail of these explanations is that the agent cannot make money on the stock market. While assertions about the strength of gravity can be asserted without evidence, assertions about the capabilities of the agent cannot be. In this chapter, we aim to remedy this dichotomy.

1.2.1 The Difficulty of Formalizing Self-Referential Claims

The difficulty of incorporating self-referential claims into the traditional Bayesian framework comes from the fact that self-referential claims allow the agent to use their own proposed strategy as information about the world. In essence, this runs into the same ambiguity present in the debate between CDT and EDT. Worse, while the distinction between CDT and EDT largely occurs in situations where there is a shared causal ancestor between the agent and the world, self-referential beliefs extend this ambiguity to situations where the strategy and the world are uncorrelated. That means that while traditional disagreements between CDT and EDT often occur in the presence of clones, perfect simulations, or memory loss, these same disagreements can occur whenever a claim is made about the consequences of one's actions. As such claims are made in everyday situations, for instance "I am going to fail this math test", self-referential claims extend this philosophical debate to a much broader range of circumstances.

The fundamental ambiguity comes from the fact that while the self-referential is clear factually, it is ambiguous what the agent should believe counterfactually. That is when the agent considers taking a different strategy than the one they factually take, should they imagine the self-referential claim applying to the new strategy or the counterfactual strategy? For instance, if one is in a prisoner's dilemma and it is claimed that "the other player will play the same strategy as you" and you consider an alternative strategy, what should you believe? Should you believe that the other person's strategy would change, under the logic that they would still play the same strategy as you, or should you believe that the other person's strategy would remain fixed, under the logic that they do what you do factually? In the prisoner's dilemma, this difference actually matters the former produces agents which cooperate, and the latter produces agents which defect.

We formalize each of these approaches, the first approach, which believes the self-referential claim will apply to the counterfactual strategy, we call the agent-referential interpretation, as the claim applies to the agent itself. The second approach we call the strategy-referential interpretation, claiming instead that the self-referential claim will apply to the factual strategy rather than the counterfactual strategy.

A keen observer would note that the definition of strategy-referential claims is actually circular. The strategy is used to interpret the claim, which defines the belief, which itself is used to justify the claim. One may be concerned that this circularity could cause strategy-referential claims to be ill-defined. With Theorem 4 we show that in finite settings as long as the strategy-referential belief is closed, the fixed point always exists. Thus, for most reasonable self-referential beliefs the strategy-referential interpretation is consistent.

Moreover, while the example of the prisoner's dilemma shows that the two interpretations of self-referential claims can lead to distinct results, we show in Theorem 5 that this is one of a small number of edge cases. In fact, the agent-referential interpretation often results in behavior consistent with the strategy-referential interpretation, and thus much of the time the difference between them is insignificant.

Finally, we offer a detailed analysis of strategy-referential beliefs in several cases and show a number of nuances arising from the circularity of their definition. These include the existence of self-referential belief, equilibrium selection problems, and stubborn agents which cannot be made to believe a strategy-referential claim despite it being consistent to do so.

Taken together, these results imply that one can consistently and unambiguously use self-referential claims in a wide range of settings without resolving all of the philosophical issues involved. However, one must be careful of the special cases listed in Theorem 5, and, if one is using the strategy-referential interpretation, one must be careful of a number of counterintuitive edge cases. This lays the groundwork for the study of self-referential claims in practical settings and motivates preferring the simpler agent-referential interpretation when the ambiguity between interpretations is known not to matter.

1.3 Solving Problem Specifications via Unsupervised Environment Design

Finally, in Chapter 4, We show how we can build scalable approaches to solve the problem specifications discussed in the previous section, by generating adaptive distributions of environments. We formalize the problem of automatically producing adaptive distributions of training environments as Unsupervised environment Design and show that the framework is expressive enough to be able to produce behavior consistent with any decision theory under the ignorance that we know of.

To show that this approach can work stably, we present an algorithm that can find policies that minimize the worst-case expected regret as the equilibrium of a 3-agent system. This algorithm, Protagonist Antagonist Induced Regret Environment Design (PAIRED). In addition to the transfer benefits that one would expect from the theory of problem specification, it also achieves a useful curriculum of increasingly complex levels. This transfer performance and complexity serve to validate the theory of problem specification. We show that, in settings where there is a well-defined notion of success that determines the majority of the reward, a minimax regret specification achieves low specification error and thus should have good transfer performance. In essence, the minimax regret specification acting to "throw out" much of the complexity of human-generated environments, so it is expected that an algorithm generating environments to fit that specification would have to "fill the complexity back in" in the same way that the forgotten information in a JPEG compression is "filled in" by the decompression algorithm.

1.4 Detailed Breakdown of Contributions

- Specification Design Problems
  - We formalize Specification Design Problems as a formal model of the decision faced by the designer when the problem specification paradigm.

  - We introduce *specification error* as the amount of the utility lost by the designer by simply specifying the wrong problem.
  - We show that specification error can be decomposed into underspecification error and misspecification error.
  - We show how underspecification error can be analyzed by constructing a measure of the value of information called specification value.
  - We show how misspecification error can be analyzed geometrically by considering the curvature of the Pareto frontier
  - We demonstrate how this analysis of specification error can be used to determine the optimal length of a specification for the designer, demonstrating a phenomenon similar to the bias-variance trade-off.

- Self-Referential Claims
  - We show that self-referential claims are necessary for the efficient specification of certain tasks, thus a Bayesian AI designer in a specification design problem will often choose to design non-Bayesian AI Systems
  - We formally define self-referential claims as claims made jointly about the strategy and the environment, such as "you will be unable to fix the engine".
  - We show that self-referential claims allow the agent to use their strategy as information about the world even when the two are uncorrelated because the self-referential claim forms a collider.
  - We propose two ways of extending the Bayesian model to allow for direct specification, or updating on, self-referential claims – the strategy referential and agent referential interpretations – which are analogous to the divide between CDT and EDT.
  - We show that despite their circular definition, strategy referential beliefs are consistent as long as the self-referential claim is closed.
  - We show that, despite the differences between the strategy and agent referential interpretations, the agent referential interpretation will choose behaviors consistent with the strategy referential interpretation in a wide range of contexts.
  - We show a number of nuances arising from the circularity of strategy referential claims, including the existence of self-referential belief, equilibrium selection problems, and stubborn agents which cannot be made to believe a strategy referential claim despite it being consistent to do so.

- Unsupervised Environment Design
  - We formalize unsupervised environment design, as the problem of automatically choosing parameters for an underspecified POMDP.

- We show that the set of possible approaches to unsupervised environment design would, at equilibrium, correspond to any possible decision theory for problems of decisions under ignorance
- We propose Protagonist Antagonist Induced Regret Environment Design (PAIRED).
- We show that PAIRED can find minimax regret policies at Nash Equilibrium, that it can generate a natural automatic curriculum of increasing complexity, and show that it empirically improves transfer performance to a set of complex human-designed levels.

Chapter 2

Specification Design Problems

2.1 Introduction

Over the past several decades we have gotten very good at solving problems that can be made formally explicit. However, as AI systems interact more with vague and complex real-world systems we are finding that it is increasingly difficult to write explicit problem specifications whose solutions solve our intended real-world problem.

This is especially concerning, as explicit problem specifications are critical to how we design AI systems. As AI practitioners we operate in what I call the "specification paradigm", which we break down the process of applying an AI system to a real-world problem into two steps: specification and optimization. In the first step, specification, we translate the vague and complex real-world problem, such as getting an agent to walk across a room, to a concrete explicit problem specification, such as finding the optimal policy for an MDP. Such a problem specification takes the form of a mathematical equation, possibly with reference to a simulator or a dataset, such that solving the problem is only a matter of computation. In the second step, optimization, we take that explicit specification as the ground-truth definition of success and optimize that specification using some AI techniques such as reinforcement learning to arrive at a policy that solves our explicit specification.

The vast majority of the field focuses on this second step, optimizing the explicit specification. In this work, we will take the unusual position of assuming that work has been, or can be, done. Thus we will focus our attention completely on the first step – specification. Our aim is to better understand how to construct explicit problem specifications whose solutions solve our complex and vague real-world problems.

In fact, many of the problems facing the field, from AI Safety [24, 1], transfer [17, 32, 51], exploration [13], interpretability [39], fairness [43] revolve around an inability to specify something important accurately. In many of these fields, breakthroughs are made by proposing different problem specifications such as providing novel models of the problem, intrinsic rewards, or dataset augmentations. However, the formal tools to compare specifications are limited. Given two specifications, neither of which is exactly correct, it is currently difficult to tell, much less argue, which is better.

For instance, consider Robust MDPs [77] as an alternative to MDPs [7] to address the sensitivity of planning to minor misspecification errors in the transition matrix. Since there has yet to be an agreed-upon methodology for comparing problem specifications themselves – arguing that Robust MDPs are better or worse than MDPs relies on informal argumentation – arguing that small errors specification are likely to be costly, or the added conservatism is likely to be costly. It is often not clear what the "goal" of the argument is. If one side were to formally prove that their preferred specification is better, what is the formal statement that they would be proving? As such, disagreements around which problem specification ought to be used are often stubborn and apparently immune to argument.

In this work, we aim to take a first step towards addressing such disagreements, by formalizing the specification design problem. In essence, we claim that a problem specification is good to the extent that it serves the interests of the AI designer. That is, a problem specification is instrumental to the designer's goals. In this way, previous discussions which have had no shared framework in which to judge the value of a specification – as a specification is usually taken as the definition of value – can make formal arguments that their proposed specification ought to be preferred by showing it is useful instrumentally to the AI designer specifying their own objective. For instance, a Robust MDP may be able to be argued as better than an MDP in some contexts, by formally showing it is more useful to the designer in some contexts.

Importantly, the focus on the needs of the AI designer is not to discount the critical concerns of the effects of the system on others, it will be important to extend our conception of a "good" specification to consider others affected by the system. Strictly as a pragmatic simplification, we will restrict our focus for this work to the interests of the AI designer. This is done only as a stepping stone, to understanding how to design a system that effectively serves an individual before addressing the more challenging tasks of serving a community or a society.

Specification design problems, described formally in Section 2.3, motivate a focus on the "specification error" of an explicit specification P relative to a true specification $\tilde{w}^T$, that is, the amount of error incurred by specifying the problem itself. Given the complexity of the world, incurring specification error is unavoidable, as one cannot specify the universe exactly. However, by designing our specifications carefully, we can manage specification errors, and create specifications that will produce useful behavior anyway, even if they are not exactly correct.

In essence, writing down a specification is a form of lossy compression. To make this clear, in Figure 2.1 the steps of a specification design problem have been shown side by side with the steps of jpeg compression. In a specification design problem, the message being communicated is the true problem specification, which is compressed lossily, leaving out the less important details, as the explicit specification. In JPEG compression the message being communicated is a true image, say of a cat, which is then compressed lossily, leaving out the less important details, as a compressed picture of a cat. In both cases, the quality of the compression is judged by its ability to have the desired effect, which we call the specification

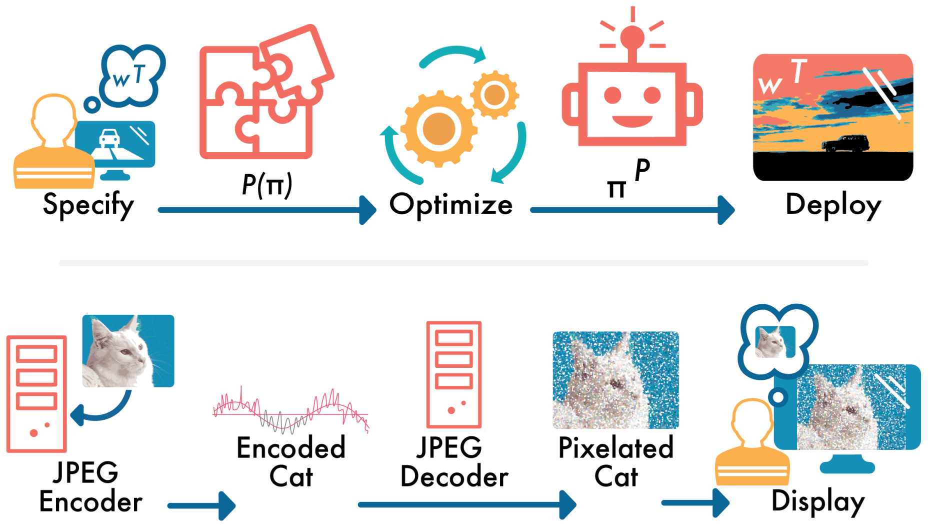


Figure 2.1: A depiction of a specification design problem (top), alongside the lossy communication problem faced by JPEG compression(bottom). Note the similarities in how, in each case, the least important information is discarded and filled in with something which achieves a similar effect to the uncompressed message.

error, either solving the problem of the designer or giving the human observer an accurate depiction of the original cat. The great success of jpeg compression is in its ability to throw away the vast majority of the information in the original image without the human observer noticing. The goal of a specification design problem is to replicate this success, leaving out the vast majority of the information about the true problem, without compromising the performance of the resulting system.

Our analysis of specification error starts by showing in Section 2.3 that specification error can be decomposed into two quantities: the errors from omitting information, which we call underspecification error, and the errors from incorrectly specifying information, which we call misspecification error.

We describe underspecification error in Section 2.5, and, taking motivation from communication theory, show that it can be characterized by specification value, a measure of the value of information in a particular claim. We show a number of interesting measures of specification value and show that the relationship between these measures can be understood through the use of Ven diagrams, in the same way as measures of information like entropy or mutual information. We describe misspecification error in Section 2.6 and characterize using a geometrical analysis of the Pareto frontier. Specifically, we find that an increased curvature of the Pareto frontier decreases the misspecification error. We show how these can be combined to provide an understanding of specification error overall, in Section 2.7, and show how the utility of the designer changes as the specification gets longer discovering a phenomenon similar to the bias-variance trade-off and similar to rate-distortion curves used in understanding the performance of JPEG compression.

Finally, to motivate our ability to characterize specification error even in environments that are dramatically underspecified, consider the following example.

Broken Brakes Example Suppose you are driving your car down a hill, and the breaks give out. The car is currently going slow but accelerating as it picks up speed moving down the hill. At the bottom, there is a cliff toward certain death. The only ways to stop are to veer into a wall in 200 meters, or in 800 meters. What do you do?

Note that this is an incredibly underspecified problem. You do not know the speed of the car, the safety systems in the car, how much damage it would incur, or how much damage you would incur. If modeled as a decision problem, we would not know any of the payoffs in the decision matrix and thus, naively, it could appear that there is way to make a decision. However, despite this extreme ambiguity but most would agree you should run into the first wall immediately. This is because this choice dominates the others – the car will only get faster over time, so running into the first wall is no worse than running into the second wall, which is definitely better than falling off the cliff.

This example demonstrates that we can know a specification, even one that has left out most of the information about the true problem, can have low specification error. In essence, even though it is underspecified, it has no underspecification error because the information that is unspecified is not necessary to make the optimal decision, even if it would be necessary to know the returns from that decision. It also shows how leaving a specification underspecified is often beneficial. The underspecification in this example results in no specification error, so aiming to specify additional information could only risk misspecification without benefit.

2.2 Related Work

Problem specifications have been modeled in the specific case of designing optimal reward functions [72] and it has been used to study how one can pragmatically interpret the specification of training environments as reveling the designer's intentions [25]. Specification design problems can be seen as a generalization of this line of research, to include the environment and decision theory along with the reward function.

There have also been several works using an information-theoretic perspective in AI to bridge the gap between a human's preferences and the agent's behavior [57, 58, 56]. However, these works use an information-theoretic perspective to build a particular fixed problem specification, whereas our approach aims to use an information-theoretic perspective to understand the choice between problem specifications.

Value of information is a widely studied concept, first discussed by Howard [29]. However, the most widely used definitions of the value of information are in terms of the value received by a Bayesian agent, requiring the specification of a prior. As the specification of the prior itself is something we would want to analyze, such approaches would result in an infinite regress. Nonetheless, these approaches serve as motivation for or notion of specification value discussed in Section 2.5.

The notion of a belief being underspecified has been studied before under the name "decisions under ignorance" as studied by Milnor [46] and described in Peterson [53], and is also quite a broad field both philosophically and practically being used across several disciplines. We largely sidestep these questions, as how the decision under ignorance is handled by the optimization algorithm is part of the optimizer and the grounding function which we assume to be given. Studying how the decision theory used by the optimization algorithm to deal with ignorance affects the misspecificaiton and underspecification errors is left to future work.

The trade-off between rate and distortion was first analyzed in Shannon's classic work [69]. Such curves show the trade-offs inherent in lossy communication, where you can reduce the accuracy of the message for an increase in communication efficiency. This idea serves as the motivation for the length-value curves analyzed in Section 2.7.

2.3 Specification Design Problems

Formally, a specification design problem is a tuple $\langle A, \Theta, U : A \times \Theta \to \mathbb{R}, \tilde{w}^T \in \Delta(\Theta), \mathcal{X}, \mathit{opt} : \mathcal{X} \to \Delta(A)\rangle$, where A is the set of possible actions, Θ is the set of unknown and unspecified parameters, U is the utility function, $\tilde{w}^T$ is the true distribution of parameters, $\mathcal{X}$ is the set of available problem specifications, and *opt* is the solution method that solves a given problem specification to given a police $\pi \in \Delta(A)$. We assume that the $A, \Theta, U, \mathcal{X}$, and *opt* are fixed and common knowledge and $\tilde{w}^T$ is known by the designer but not the optimization algorithm. The goal for the designer is, knowing $\tilde{w}^T$, to pick the specification $P \in \mathcal{X}$ to maximize $U^T(P) = \underset{\theta \sim \tilde{w}^T}{\mathbf{E}}[U(\mathit{opt}(P), \theta)]$. That is, the designer uses the specification P instrumentally to achieve their own ends.

We will judge how well the designer does this by consider the specification error, that is the difference between the utility the designer obtained from the policy optimal for P π^P and the utility the designer would have obtained from the optimal policy on the true world π^*, or formally:

$$SE(P, \tilde{w}^T) = \underset{\theta \sim \tilde{w}^T}{\mathbb{E}}[U_\theta(\pi^*)] - \underset{\theta \sim \tilde{w}^T}{\mathbb{E}}[U_\theta(\pi^P)]$$

It is clear that minimizing specification error is the same as maximizing designer utility, as the term $\underset{\theta \sim \tilde{w}^T}{\mathbb{E}}[U_\theta(\pi^*)]$ is constant with respect to the designer's specification.

2.3.1 Grounding the Language

In order to assist our analysis we will make the additional assumption that the language of available problem specifications has semantics. That is, there is some set of claims that each specification makes about the distribution of worlds $\tilde{w}$. This grounding allows us to analyze the difference between a specification being "wrong", in that it makes a claim about the true problem which is false, and a specification being "incomplete" in that it leaves out information about the true problem, but makes no false claims. Since there is a significant difference between providing someone with false information about the task and simply omitting information about the task, it makes intuitive sense that it may be useful to distinguish these two cases.

Formally, we can say that a feature of an environment is determined by some function of the parameters of the true distribution of worlds $f : \tilde{w} \to Z$, where Z is the set of possible values that feature could take on. Examples of features in our broken breaks example includes the speed of the car, the slope of the hill, or the likely damage in the case of an accident. A claim is then an assertion about the state of that feature, so one could claim that the estimated average speed of the car is $30mph$. The set of claims that are made by a specification P will thus be a set of features and their associated values, $ground(P) = (f_1, z_1), (f_2, z_2), \ldots, (f_n, z_n)$. We will assume that this map is bijective, in that a set of claims $\hat{X} = (f_1, z_1), (f_2, z_2), \ldots, (f_n, z_n)$ also implies a specification, which we will notate as $P^{\hat{X}}$ such that $ground(P^{\hat{X}}) = \hat{X}$.

Given the true world $\tilde{w}^T$ we will say that a particular claim $X \subseteq \Delta(\Theta)$ is accurate if it contains the true world $\tilde{w}^T$.

Finally, using this division, we can consider a "corrected specification", which considers a modification of some specification P to correct all of the false claims in the original specification. If the original claims are $ground(P) = (f_1, z_1), (f_2, z_2), \ldots, (f_n, z_n) = \hat{X}$, then the corrected claims are $f_1, f_1(\tilde{w}^T)), (f_2, f_2(\tilde{w}^T)), \ldots, (f_n, f_n(\tilde{w}^T))$, which we call $\hat{X}^\perp$ which gives a corrected specification $P^{\hat{X}^\perp}$, which we will notate as $P^\perp$ for simplicity. Thus by comparing the true specification $\tilde{w}^T$, the explicit problem specification P and the corrected problem specification $P^\perp$, we can study the errors coming from wrong claims and incorrect claims separately.

2.4 Decomposing Specification Error

Having assumed that the designer is solving a specification design problem with a grounded specification language, we can now develop strong tools for understanding the specification error of each specification available to the designer, and thus for helping the designer decide between specifications. The first observation is that we can decompose the specification error by dealing separately with errors arising from incorrect claims in the specification, which we call misspecification errors, and errors resulting from claims omitted from the specification, which we call underspecification errors.

To quantify misspecification errors we can consider the difference between the performance of the explicit specification P and the performance of the corrected specification $P^\perp$. Or more formally:

$$MSE(P, \tilde{w}^T) = \mathop{\mathbb{E}}_{\theta \sim \tilde{w}^T}[U_\theta(\pi^\perp)] - \mathop{\mathbb{E}}_{\theta \sim \tilde{w}^T}[U_\theta(\pi^P)].$$

For instance, in our broken brakes example. If the agent incorrectly believed that they would be able to stop before the end of the cliff, then they would incur a high misspecification error since the incorrect belief would have the effect of causing the car to careen off the cliff and incur high loss compared to the care which pulls off of the road. On the other hand, if the agent only has incorrect beliefs about the speed of the car, then this would incur a low misspecification error, as regardless of the speed the agent would pull off the road as soon as possible.

To quantify underspecification errors, we can consider the difference between the performance of the corrected problem specification and the true specification, that is:

$$USE(P, \tilde{w}^T) = \mathop{\mathbb{E}}_{\theta \sim \tilde{w}^T}[U_\theta(\pi^*)] - \mathop{\mathbb{E}}_{\theta \sim \tilde{w}^T}[U_\theta(\pi^\perp)].$$

For instance, in our broken brakes example, though the speed of the car was never specified, it is not important to decision-making, so there is no underspecification error. If instead, it was not specified if the breaks work or not, then this could lead to a high underspecification error if the agent acts in a way that would careen off the cliff. Note that in such a case, every agent must incur a high underspecification error with respect to either the case where the breaks work or the case where the breaks do not work.

The sum of misspecification error and underspecification error gives the total specification error. Thus, if we can understand misspecification and underspecification separately, and build techniques to bound the misspecification and underspecification error of a specification, we can understand and bound the specification error itself. In the next two sections, Section 2.5 and Section 2.6, we will build tools to study underspecification error and misspecification error. To understand underspecification error, we will show that we can construct a notion analogous to the value of information, which quantifies the value of an assumption in decreasing the specification value, as well as allowing us to understand how the value of an assumption may depend on what other assumptions are also being made. To understand misspecification error, we will show how misspecification can sometimes be analyzed geometrically, and how small changes in the beliefs of the agent affect the performance of the chosen strategy, by noting the curvature of the Pareto frontier.

2.5 Underspecification Error as measure of the Value of Information

When we analyze underspecification errors, we do not have to worry about the possibility of misspecification. Thus we only have to be concerned with which facts would be most useful to specify to cause the resulting policy to perform well on the true objective.

When specifying a problem there are many facts that one may consider specifying. The real world is a big and complex place, with virtually limitless detail to explain. However, for any particular problem, the vast majority of these details do not matter at all.

For instance, when I am specifying the problem of getting a humanoid robot to walk across the room, it is important to specify in some form something about there being a 3-d environment, how friction works, how gravity works, and how torques correspond to limb actuation. However, it would be useless to specify the positions and colors of all of the bicycles in France. More to the point, there are many partially relevant aspects of the environment, such as the texture and plasticity of the floor, the aerodynamics of the robot and the air-pressure of the room, or the humidity and thus the possibility for condensation changing friction coefficients. There is this a general sense that, at some level of resolution, it should be okay to stop without loosing too much performance, that some facts are relatively less important than others. In this section we aim to move towards a formalism of that sense.

A natural idea that may come to mind when considering the relative value of specifying different features of the environment is the Bayesian value of information, which would allow one to calculate how much performance a Bayesian agent would gain from learning each of these facts. However, it is difficult to directly apply the Bayesian notion of value of information in this context, since it would need to be calculated with respect to a prior which has also not been specified. Thus this results in a regress which, in order to calculate the value of a particular specification, one would first have to specify a prior over all specifications, which itself is difficult to specify.

Our approach instead builds on a pragmatic and instrumental interpretation of the value of an assumption. That is, rather than defining the value of an assumption with respect to some prior over specifications, we instead define the value of the assumption in terms of the degree to causes the policy to more effectively optimize the designer's utility.

Concretely, we can define the specification value of some claims $\hat{X}$, relative to specification design problem with grounded semantics, as the difference in the performance of the problem specification with that assumption and without that assumption. Or concretely:

$$SV(\hat{X}) = Value(\hat{X}) - Value(\emptyset)$$

Where $Value(\hat{X}) = \mathop{\mathbb{E}}_{\theta \sim \tilde{w}^T}[U_\theta(opt(P^{\hat{X}})]$, is the utility of the designer after specifying claims $\hat{X}$.

For instance, consider the problem of taking a 2 question true or false test, where question 1 is determined by $\hat{X}$ and question 2 is determined by $\hat{Y}$ the test is scored as follows
Test 1:

$$U_\theta(\pi) = 10\,\mathbf{P}(\text{Q1 Correct}) + 3\,\mathbf{P}(\text{Q2 Correct}).$$

That is, the policy gets 10 points for getting the first question right and 3 points for getting the second question right. In this case, the specification value of specifying the answer to the first question would be 5, since the policy would otherwise have a 50% chance of getting it correct by chance, and the specification value of specifying the second answer would be 1.5 likewise.

Specification value is quite simple for understanding the value of a particular claim in isolation. However, often times two claims can contain effectively the same information, and

thus their value cannot be measured independently. For example, consider the case where the agent is taking a quiz where getting either question right gives 10 points as follows:
Test 2:

$$U_\theta(\pi) = 10\ (\mathbf{P}(\text{Q1 Correct}) \vee \mathbf{P}(\text{Q2 Correct}))$$

In this case, the specification value for specifying the answer to question 2 decreases to 0 if the agent already knows the answer to question one, as they can already get all of the available points.

To account for these situations, one could define the *relative specification value* of an assumption $\hat{X}$ relative to $\hat{Y}$. Now measuring the change in the value caused by $\hat{X}$ when the claim $\hat{Y}$ is already known. Or formally,

$$RSV(\hat{X}|\hat{Y}) = Value(\hat{X} \wedge \hat{Y}) - Value(\hat{Y}).$$

When analyzing our "Test 2" example above, we find that the relative specification value of each variable relative to the other is 0 as expected, or formally $RSV(\hat{X}|\hat{Y}) = 0$ and $RSV(\hat{Y}|\hat{X}) = 0$. It is also useful to note at this point that $USE(P^{\hat{X}}) = RSV(\hat{\tilde{w}^T}|\hat{X})$, where $\hat{\tilde{w}^T}$ is the set of all truthful claims that can be made about $\tilde{w}^T$. That is, the underspecification error is the relative value of knowing the true specification exactly conditioned on knowing only the claims contained in the true specification.

Returning to the "Test 2" example one may think that, in some sense the valuable information is "shared" between $\hat{X}$ and $\hat{Y}$. In some sense they contain "the same" valuable information. In fact we can make that intuition formal and define the *shared specification value* between $\hat{X}$ and $\hat{Y}$ to be the value of knowing that either claim is true, but not knowing which (or both) is true, or formally:

$$SSV(\hat{X};\hat{Y}) = SV(\hat{X} \vee \hat{Y})$$

In this case, the $SSV(\hat{X};\hat{Y})$ is 2.5, as random guessing both answers will give 7.5, and knowing a plausible answer for either will give a guaranteed 10.

One can also consider the opposite, to pieces of information which are only useful if both are known. For instance, a test where you must get two questions both right in order to get any points: **Test 3:**

$$U_\theta(\pi) = 10\ (\mathbf{P}(\text{Q1 Correct}) \wedge \mathbf{P}(\text{Q2 Correct}))$$

In this case, the relative value of $\hat{X}$ given $\hat{Y}$ increases, since without knowing $\hat{Y}$ there is a 50% chance of missing that question, cutting any utility that you would get from getting question 1 correct in half. In some sense, this value is distributed across the two claims. That is, it is the value that you get from both claims that you would not get from either claim individually.

Indeed, we can make this intuitive concept formal as well by defining the *distributed specification value*, which measures how much more valuable $\hat{X}$ is when $\hat{Y}$ is known, rather than if just $\hat{X} \vee \hat{Y}$ were known, or formally:

$$DSV(\hat{X};\hat{Y}) = RSV(\hat{X}|\hat{Y}) - RSV(\hat{X}|\hat{X} \vee \hat{Y})$$

In the case of "Test 3", the distributed specification value between $\hat{X}$ and $\hat{Y}$ is $DSV(\hat{X};\hat{Y}) = RSV(\hat{X}|\hat{Y}) - RSV(\hat{X}|\hat{X} \vee \hat{Y}) = 5 - \frac{5}{3}$. One could worry that this definition would change if we defined it in terms of the usefulness of $\hat{Y}$ as opposed to $\hat{X}$, that is the equation isn't obviously symmetric in $\hat{X}$ and $\hat{Y}$, though the distributed value ought to be. Despite the apparent asymmetry of the definition, the distributed specification value is, in fact, symmetric.

Theorem 1. *The distributed specification value is symmetric. That is, for all X, Y we have $DSV(X;Y) = DSV(Y;X)$*

Proof. By definition we have:

$$\begin{aligned}
DSV(X;Y) &= RSV(X|Y) - RSV(X|X \vee Y) && (2.1)\\
&= Value(X \wedge Y) - Value(Y) && (2.2)\\
&- Value(X) + Value(X \vee Y) && (2.3)\\
&= RSV(Y|X) - RSV(Y|Y \vee X) && (2.4)\\
&= DSV(Y;X). && (2.5)
\end{aligned}$$

Thus distributed specification value is symmetric as desired. □

When considering the interactions between the value of two claims $\hat{X}$ and $\hat{Y}$, we would want to account for both the elimination of shared valuable information and the addition of distributed valuable information. To do this we can can define the *entangled specification value.* Formally, this can be defined as the value contained in $\hat{X} \vee \hat{Y}$ which is not in $\hat{X}$ given $\hat{Y}$ or $\hat{Y}$ given $\hat{X}$. Or formally,

$$ESV(\hat{X};\hat{Y}) = SV(\hat{X} \wedge \hat{Y}) - RSV(\hat{X}|\hat{Y}) - RSV(\hat{Y}|\hat{X})$$

In fact, we can show that the entangled specification value is the shared value minus the distributed value.

Theorem 2. *The entangled specification value is the shared value minus the distributed value. Or formally, for all X, Y we have $SSV(X;Y) - DSV(X;Y) = ESV(X;Y)$.*

Proof. By definition we have:

$$\begin{aligned}
SSV(X;Y) - DSV(X;Y) &= SV(X \vee Y) - RSV(X|Y) + RSV(X|X \vee Y) && (2.6)\\
&= Value(X \vee Y) - Value(\emptyset) && (2.7)\\
&- Value(X \wedge Y) + Value(Y) && (2.8)\\
&+ Value(X) - Value(X \vee Y) && (2.9)\\
&= SV(X) - RSV(X|Y) && (2.10)\\
&= ESV(X;Y). && (2.11)
\end{aligned}$$

Where the last line follows from Theorem 3. Thus we have shown that the entangled value is the difference between the shared value and the distributed value, as desired. □

Though we have just independently defined several ways of comparing the information in various assumptions, these measures are all closely related. In fact, the relationship between specification value, relative specification value, and shared specification value are analogous to that between entropy, conditional entropy, and mutual information. In the next section, we will show that many of the same relationships between these terms apply.

2.5.1 Specification Value as a Signed Measure

In this section, we will show that, like other measures of information, the discussed measures of specification value act as a signed measure. Thus, in the same way as other measures of information, you can intuitively understand the interaction between these definitions through Ven diagrams.

Consider the Ven diagram shown in Figure 2.2. Intuitively, you can imagine the specification value of $\hat{X}$ and $\hat{Y}$ as the area of the red and blue circles respectively. The area of the red circle not covered by the blue circle represents the relative specification value of $\hat{X}$ given $\hat{Y}$, which is intuitively the value of knowing $\hat{X}$ given you already know $\hat{Y}$. Similarly, the area of the blue circle which is not in the red circle corresponds to the relative specification value of $\hat{Y}$ given $\hat{X}$, that is intuitively the value of knowing $\hat{Y}$ given that you already know $\hat{X}$.

The purple intersection of the two circles is the entangled value of $\hat{X}$ and $\hat{Y}$, that is, how much knowing one would influence the value of the other. Since the entangled specification value is symmetric, $ESV(\hat{X};\hat{Y}) = ESV(\hat{X};\hat{Y})$, it intuitively makes sense as a well-defined region on this graph. Taken together, the entirety of the overlap of both circles corresponds to the specification value of both assumptions together $SV(\hat{X} \wedge \hat{Y})$, as it would be all of the value in either of the claims $\hat{X}$ or $\hat{Y}$.

Intuitively, if this picture is a correct conceptualization of the relationship between the concepts representing each region, we should be able to infer a number of relationships between these concepts. Most straightforwardly the definition of entangled specification value follows from noting that the full area of the purple region is the full area of the ven diagram minus the red and blue regions, or translated into an equation $ESV(\hat{X},\hat{Y}) = SV(\hat{X} \wedge \hat{Y}) - RSV(\hat{X}|\hat{Y}) - RSV(\hat{Y}|\hat{X})$ as in the definition.

Beyond recovering this definition, we can discover many other useful relations. For example, the specification value of $\hat{X}$ relative to $\hat{Y}$ plus the specification that $\hat{X}$ has which is entangled with $\hat{Y}$, recovers the specification value of $\hat{X}$ itself.

Theorem 3. *The relative specification value of $\hat{X}$ given $\hat{Y}$ plus the entangled specification value of $\hat{X}$ and $\hat{Y}$ is the specification value of $\hat{X}$. That is, $RSV(\hat{X}|\hat{Y}) + ESV(\hat{X};\hat{Y}) = SV(\hat{X})$.*

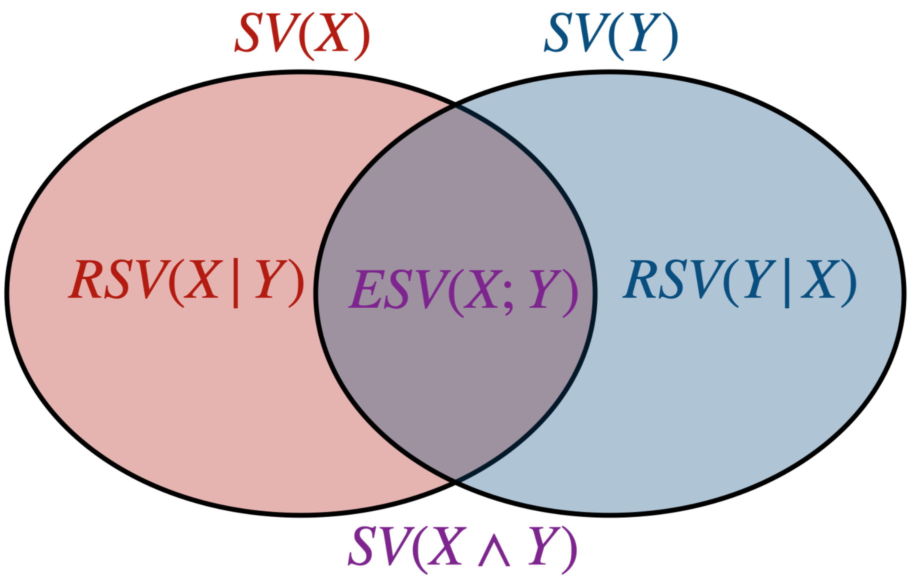


Figure 2.2: A Ven Diagram of specification Value Measures. Each region of the diagram is marked with a corresponding measure. It can be shown that the measures corresponding to larger regions are the sum of the measures of the component regions.

Proof. By definition we have:

$$RSV(\hat{X}|\hat{Y}) + ESV(\hat{X};\hat{Y}) = Value(\hat{X} \vee \hat{Y}) - Value(\hat{Y}) \tag{2.12}$$

$$+ Value(\hat{X} \wedge \hat{Y}) - Value(\emptyset) \tag{2.13}$$

$$- Value(\hat{X} \wedge \hat{Y}) + Value(\hat{Y}) \tag{2.14}$$

$$- Value(\hat{X} \wedge \hat{Y}) + Value(\hat{X}) \tag{2.15}$$

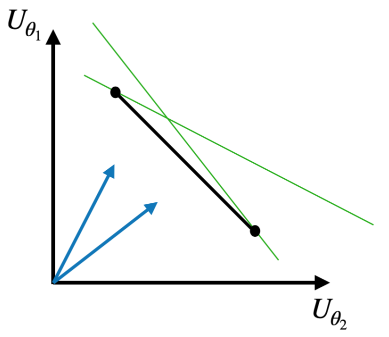
$$= SV(\hat{X}). \tag{2.16}$$

Thus the specification value of $\hat{X}$ can be written as the sum of the specification value of $\hat{X}$ relative to $\hat{Y}$ and the entangled value of $\hat{X}$ and $\hat{Y}$. □

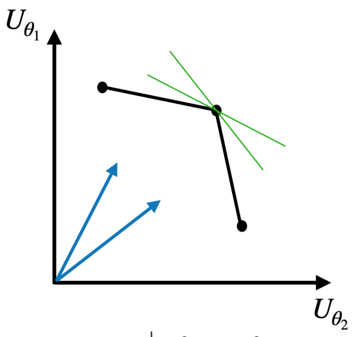
Thus not only do these measures of information relate to the specification error of the designer, but they intuitively represent signed measures that can be understood to behave similarly to a measure of area.

2.6 The Geometry of Misspecification Error

Unlike underspecification, misspecification cannot as easily be conceptualized as a value of information, since the information being provided is inaccurate. Instead, we propose a



	θ_1	θ_2
a_1	10	0
a_2	0	10



	θ_1	θ_2
a_1	10	0
a_2	0	10
a_3	9	9

Figure 2.3: The payoff matrix and Pareto Curves for "guess the bit"(left) and "pay to guess the bit"(right) problems respectively

different method of characterizing misspecification error modeling the problem geometrically.

The motivation for this method comes from the idea of the Pareto frontier, which allows one to visualize the trade-offs between a set of different objectives by considering the policies which are not dominated. That is the policy for which there is no other policy outperforms it on every objective. In two dimensions, these plots can be visualized as a graph showing the amount of utility achievable by objective A while achieving a certain value for objective B. As shown in Figure 2.3. The set of policies shown in this figure is called the "Pareto Frontier" and this frontier characterizes the trade-off between the two objectives.

Thinking of the performance on each deterministic parameterization θ as its own objective, the Pareto Frontier across the performance on these worlds characterizes the fundamental trade-offs present when selecting policies that will be deployed in one of these worlds.

Figure 2.3 shows two problems and their corresponding Pareto curves. In the first of these problems "guess the bit", there is a strong trade-off between performance in θ_1 and θ_2, such that any performance gained in 1 is lost in the other. This results in a Pareto frontier that does not have much curvature. Alternatively, the second problem "pay to guess the bit", adds an option to receive most of the possible reward by choosing a safe action a_3. This has the effect of extending the Pareto frontier up and to the right, resulting in a Pareto frontier with more curvature.

Now consider a true world near the uniform distribution between θ_1 and θ_2, $\tilde{w}^T = \{\theta_1 : 50\%, \theta_1 : 50\%\}$. The optimal policy for a world can be visualized as the point on the Pareto frontier with a tangent line whose norm corresponds to the probability distribution $\tilde{w}^T$. We have visualized each of these tangent lines in the figure using green lines.

In "guess the bit", any small change to the probability distribution, slightly increasing or decreasing the probability of θ_1 will result in choosing a different action. Thus small changes in the belief can result in choosing a suboptimal action which can dramatically reduce performance. This can be seen geometrically by noticing that the green tangent lines, even though they have very similar slopes, intersect the Pareto Frontier very far apart, indicating a strong trade-off.

In "pay to guess the bit" the story is very different. Small changes in the slope of the tangent line result in no change to the action since the Pareto frontier has high curvature. As such, there must be a large deviation to be 90% confident in a certain parameterization, before you incur any misspecification error.

In fact, these features of the Pareto frontier can be made formal by modeling the Pareto Frontier itself as a geometric object. Intuitively, you can think of the Pareto Frontier as the convex hull of the values of the deterministic actions in the deterministic worlds, which we make completely precise in the next section

2.6.1 Geometry of the Pareto Frontier

The intuition behind the Geometric Pareto Representation(GPR) of problem specifications, is to analyze the Pareto frontier graph of the performance of a particular strategy with respect to each of the deterministic worlds θ as a geometric object. The aim of this is to allow us to represent problem specifications, regardless of how they are written as related geometric objects, and aim to understand their geometric properties.

We define the geometric Pareto representation in the $\mathbb{R}^{|\Theta|}$. Every world, being a probability distribution over Θ, lies on the simplex of this space. Thus each world can be interpreted as a vector with a unique direction from the origin.

To build the Pareto frontier graph, we can take the strategy space Σ which can be understood as a simplex in $\mathbb{R}^{|A|}$. And map to our space $\mathbb{R}^{|\Theta|}$ so that every strategy is placed at the coordinates corresponding to their utility on the corresponding deterministic world. Or more formally, we define the map $f : \mathbb{R}^{|A|} \to \mathbb{R}^{|\Theta|}$ to be:

$$f(\sigma)_i = \mathop{\mathbb{E}}_{a \sim \sigma}[U(a, \theta_i)].$$

Note here that we have now done something a bit weird – we have given two interpretations to the vector space $\mathbb{R}^{|\Theta|}$. In one sense, a point in this vector space, on the simplex, can be interpreted as a world where each dimension is a *probability* of the deterministic parameterization occurring in that world. In another sense, a point in this vector space can be interpreted as a strategy, where each dimension is a *value* of the strategy on each deterministic parameterization. Though this is a bit unusual, it allows for a space that is relatively intuitive and in which many properties of problem specifications that are of interest can be understood as natural geometric properties.

The first thing to note about f is that it is linear. Thus, since the simplex of strategies in $\mathbb{R}^{|A|}$ is convex and bounded, the image of $f(\Sigma)$ is convex and bounded. In general, the image of $f(\Sigma)$ can be any arbitrary convex polytope, since the performance of the deterministic strategies on the deterministic worlds determine the corners of the convex polytope and can be set independently to match the coordinates of vertices of whatever convex polytope you wish.

The second thing to note is that the utility of any strategy on a fixed probabilistic world is exactly the dot product between the strategy's value embedding and the world's probability embedding. Or breaking it out formally:

$$\begin{aligned}
U(\sigma, w) &= \sum_{\theta_i \in \Theta} \sum_{a \in A} p_w(\theta_i) p_\sigma(a) U(a, \theta_i) && (2.17)\\
&= \sum_{\theta_i \in \Theta} p_w(\theta_i) \sum_{a \in A} p_\sigma(a) U(a, \theta_i) && (2.18)\\
&= \sum_{\theta_i \in \Theta} p_w(\theta_i) f(\sigma) && (2.19)\\
&= w \cdot f(\sigma) && (2.20)
\end{aligned}$$

Given this, we can also give a more geometric interpretation – the utility of any strategy on a fixed world is also directly proportional to the length of the projection of the image of that strategy through f onto the world. Or more formally, for any fixed world $w \in \Delta(\Theta)$, there is a constant C such that, for all strategies $\sigma \in \Delta(A)$ we have:

$$U(\sigma, w) = C|proj_w(f(\sigma))|_2^2.$$

This is simply because of the relationship between the projection and the dot product, as follows:

$$\begin{aligned}
U(\sigma, w) &= w \cdot f(\sigma) && (2.21)\\
&= |w|_2^2 |proj_w(f(\sigma))|_2^2 && (2.22)
\end{aligned}$$

where $C = |w|_2^2$ is a constant for a fixed world w. This distortion factor $|w|_2^2$ would go away if we normalized the world embedding by making them unit vectors in the l_2 norm before considering the projection, in which case the size of the projection would be exactly $U(\sigma, w)$ independent of w.

Taken together, we can see that the optimal policy with respect to w will be the one whose embedding is largest when projected onto the world embedding. This is equivalent to saying that the optimal policies are those whose embedding is on the tangent line to the Pareto frontier which is perpendicular to the world embedding.

Given that, we could analyze explicitly how the curvature of the Pareto frontier corresponds to the misspecification error caused by small specification errors. The intuitive analysis we have shown here, combined with our prior analysis of underspecification error, is enough to usefully characterize specification error overall, as we will show in the next Section.

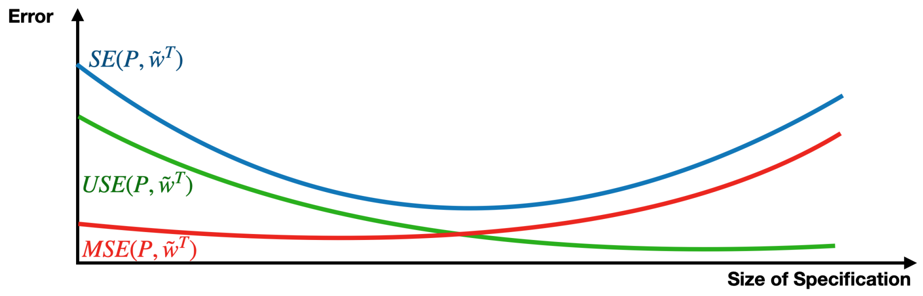


Figure 2.4: A visualization of the specification error, misspecification error, and underspecification error, as the length of the specification increases.

2.7 Optimal Specification Length

Given that the designer cannot specify all of reality, it is natural to wonder when one should stop specifying a problem. This is analogous to the problem of what details are safe to remove in JPEG compression. In this section we will study this question formally through the specification design problem framework, translating the idea of rate-distortion curves used in lossy compression theory to length-value curves in the context of specification design problems.

To formalize this problem we need some notation of specification length. To do this we can consider a sequence of specification design problems, where the only difference between each specification is the set of available specifications to choose from. In the n-th specification design problem, the designer can specify the n features among a set of available features $\mathcal{F}$. Or formally, the nth specification design problem would be defined by the tuple: $\langle A, \Theta, U : A \times \Theta \to \mathbb{R}, \tilde{w}^T \in \Delta(\Theta), \mathcal{X}^n, opt^n : \mathcal{X} \to \Delta(A) \rangle$, where $\mathcal{X}^n$ is the set of specifications which contain any n claims from the set $\mathcal{F}$, and opt^n maximizes expected value assuming a uniform random distribution over unspecified θ^n.

To analyze a concrete case, consider the specification design problem where the task to be solved is n independent instances of the C, I "pay to guess the bit", where C^n and I^n are common knowledge, as shown in Figure 2.5. The features that are available to be specified are the probability of each θ_1^n. We will be analyzing the performance of an error-prone designer so noise is added to the specification of each probability $n \sim \mathcal{N}(0, \frac{1}{4})$. If adding the noise would result in a probability below zero or above 1 it returns 0 or 1 instead to ensure what is specified is, in fact, a distribution.

To visualize how the performance of the designer changes as n varies, we visualize the optimal specification error and corresponding underspecification and misspecification error in Figure 2.4. As the designer specifies more probabilities, we can analyze the specification error of the designer.

Since each bit of the specification is referring to an independent game, the job of the designer is reduced to prioritizing which bits to specify. The bits that are best to specify would

$$\begin{array}{c|cc} & \theta_1 & \theta_2 \\ \hline a_1 & C & 0 \\ a_2 & 0 & C \\ a_3 & I & I \end{array}$$

Figure 2.5: The payoff matrix of general C, I "pay to guess the bit"

most quickly decrease the underspecification error, and most slowly increase the expected misspecification error. At first, there could be many easy wins, for bits where I^n is small and C^n is large, and θ_1^n is near 0 or 1. In such cases, there is very little risk of misspecification since, as our analysis in Section 2.6 and Figure 2.3 shows small errors in the specification do not change the policy. There is also high reduction in underspecification error, because C is large and so the correct specification has a lot of specification value as defined in 2.5

However, as the n increases there will be a decreasing marginal return on the additional bits of specification since the best specification opportunities have already been used. Eventually, the only information which has not been specified are bits with high I^n and low C^n and for which θ_1^n is near 0.5. In such cases, there is now a high risk of misspecification, since the Pareto curve is shallow, and low upside, since the difference from optimal behavior is small, due to the true distribution of parameters being near 0.5.

Surprisingly, after some point, additional bits of specification no longer help, and in fact are counterproductive, as the decrease in underspecification error is overcome by an increase in expected misspecification error. In the extreme case, there will be some bits for which the default behavior is actually optimal but the specification noise could introduce an error, so there will be no decrease in underspecification error, and the misspecification error would decrease performance.

Overall, this results in a decreasing underspecification error, an increasing misspecification error, and an overall specification error that is concave with an optimum which does not fully specify the problem. The resulting graph is reminiscent of the bias-variance trade-off, and in fact, there is a deep connection between the two if you think of the specification as a model. While these curves are generally representative it's important to note that neither underspecification error nor misspecification error are necessarily monotonic, and so specification error is not necessarily convex. However, this analysis shows that the specification design problem framework can be used to understand the optimal behavior of the designer.

2.8 Conclusion

We have introduced specification design problems as a formalization of the problem faced by an AI designer when specifying a problem for the AI system to solve. We have shown how the specification design problem can be productively analyzed by decomposing specification error into underspecification error and misspecification error. To understand underspecification error we developing specification value, as an analog to the value of information, we have given a tool for AI designers to the effectiveness of the information they specify in creating more justifiable policies. Then we described how misspecification error could be characterized geometrically by considering the curvature of the Pareto frontier. Finally, we combined these techniques to understand specification error in a concrete context. We leveraged the framework of specification design problems to analyze the optimal specification length for the designer, showing length-value curves which offer a powerful tool to understand the marginal value of specification and the optimal specification length.

These techniques are only the first to prove the value of analyzing specification design problems. The specification design problem lays the foundations for the analysis of a wide array of decision problems designers may face, from specifying optimal exploration bonuses or reward-shaping terms to specifying strategies to moderate discussions, empower communities, or align AI systems to the objectives of society.

Chapter 3

Self Referential Claims

3.1 Introduction

In order to operate in the world it is important for agents to have accurate beliefs about the consequences of their own actions. It is important for an agent to understand their own capabilities, not to be overconfident or underconfident in their ability to accomplish a particular task or outcome. An agent who is overconfident in their abilities to do death-defying stunts is likely to get hurt, one who is overconfident in their ability to navigate tricky social situations is likely to make a fool of themselves, and an agent who is overconfident in their investment ability is likely to lose a lot of money. Alternatively, an agent who is underconfident will be overly reserved and miss out on valuable opportunities. In this work, we call the beliefs that an agent has about the consequences, or likely consequences of their own actions, self-referential beliefs, and assertions about the consequences of an agent's actions, self-referential claims.

However, while self-referential beliefs are critical to agent success, it is typically an afterthought in the traditional Bayesian model of AI. While direct empirical beliefs about model parameters – beliefs about physics, other agents, and market conditions – can be, and often are, explicitly and directly specified by the AI designer, self-referential beliefs – beliefs about the likelihood to make a jump, one's ability to navigate in a social context, or one's ability to effectively invest – can usually only be derived from the agent predicting how their direct empirical beliefs interact with their planned policy. In this way, the claim that an agent is being overconfident or underconfident is treated as an afterthought.

For instance, while it can be directly asserted that acceleration due to gravity is between 9 and 10 meters per second per second, and can be used in a Bayesian update, trying to assert that an agent is overconfident in their investing ability, and is likely to lose money on the stock market, is not as easily done. In fact, we show in Section 3.2.3.1, that a natural way that this update would likely be implemented would cause the agent to change strategies but still disbelieve the asserted claim, just for different underlying empirical reasons.

In fact, even if the AI designer were absolutely certain that a particular self-referential claim was true – that an agent would be completely unable to perform some task for instance because the AI designer did not specify some requisite knowledge – in the standard framework the AI designer could not directly make the agent believe the self-referential claim, and would instead need to find some direct empirical beliefs from which the agent itself could derive that claim. Rather than telling an agent that they will be unable to make money on the stock market, the AI designer would have to construct beliefs about the market such that the agent could themself derive that they would be unable to make money. However, even this is challenging! Of course one could lie to the agent, saying that the market will respond adversarially to their actions, but if one is committed to being truthful one is then obligated to explain what it is about the stock market that makes it difficult to make money.

One attempt at such an explanation would be to reference the efficient market hypothesis – that there is not, in fact, a predictable way money can be made on the stock market as any predictable strategy has been taken. However, note that such a claim, when taken literally, is false, as other agents can and do consistently make money on the stock market. The accurate claim is that the agents with the knowledge and abilities of a retail trader, or agents like our agent, cannot predictably make money on the stock market as strategies that they could predict would work have been taken. However, this claim is now inherently self-referential, referencing the skills and abilities of the particular agent being designed, and thus cannot be directly empirically specified either. Notice that all of this difficulty is to get the agent to believe a self-referential claim that the designer already knows to be true – a more effective use of time may be to find a way to assert to the agent that the self-referential claim is in fact true and offload the work of understanding exactly why, if it is necessary to understand why, to the agent.

3.1.1 Example: Fixing an Engine

As a more concrete example, consider a situation in which an agent's car has broken down near a junkyard on the way to an important event. The agent believes that they can fix the car using spare parts, but only has time to try one fix. Alternatively, the agent could pay a mechanic to fix the car, and ensure that they make it to the event on time. The agent is dramatically overconfident in their ability to fix the engine, due to their underestimation of the complexity of modern engines, and by default would attempt to fix it themselves, which we know would fail. If giving a friend advice in such a situation it is quite natural to say "you will be unable to fix the engine", which, if they trust us, would cause the agent to pay the mechanic and get to the event on time. In Section 3.2.3.1 we show how our formal models of self-referential claims arrive at this solution.

However, notice how hard it would be to truthfully convince the agent to use the mechanic without referencing the agent's abilities, only referencing empirical facts about car engines. You could explain why the plan they have to fix the engine would not work, but they could then update that plan, which you could then again discredit, but they could again update that plan ad infinitum. You could explain that modern engines are complex and aim to explain these complexities, but they may still believe that they understand the underlying principles and thus the failure modes are simple. You could try to assert that the engine is not fixable with the tools at hand, and while that would convince your friend, it would not be true. Notice again that, throughout all of these attempts, the only important conclusion is that your friend understands that they will be unable to fix the engine. The only point of difficulty is the inability to assert, or their inability to accept, this claim directly.

3.1.2 Applications in AI

There is a saying that "One man's modus ponens is another man's modus tollens". That is when it becomes known that A implies B, it can either be taken as evidence for B or against A. In the standard Bayesian model implications from the direct empirical beliefs to self-referential claims only go one way. If the model says that you can fix the engine, then you can. Self-referential beliefs allow one to take the modes tollens of this implication – If the model says that you can fix the engine, then the model is broken. In effect, this allows for a sort of error correction, where the implausible consequences of models can be used as evidence to correct the model. Moreover, allowing modus tollens where there have only been modus ponens gives a dramatically new affordance to the AI designer, where lines of reasoning which were previously impossible to incorporate can be effectively formulated and integrated.

For instance, an AI designer may simply design their agent to know that they will be unable to make money on the stock market by explaining why. More ambitiously, a designer could make an agent which directly tracks their ability at the stock market and believes that they will be unable to make money on the stock market if their last 100 attempts to do so have failed, even if they do not know why they have failed. A similar case can be made in the multi-agent systems for calibrating confidence in opponent models. In essence, self-referential claims offer a model for self-confidence and self-doubt which is broadly applicable. In fact, models of worst-case analysis or risk aversion can be cast as self-referential claims, for instance, the claim that the world is one of the worst for you, and thus while they would not fall into a traditional Bayesian framework they can be modeled as self-referential claims.

Self-referential claims are also a boon for AI safety researchers, as often it is easier to notice a failure than to fix it. One may aim to correct overconfidence in the reward function causing ignoring direct commands [26, 45] by directly specifying an agent which believes "I am unlikely to be correct when I ignore direct commands". One may attempt to address goal misgeneralization [18] by specifying "When there were ambiguous goals in training, I am unlikely to have chosen the correct one". While such attempts are not full solutions to the problems in themselves – there is still the task of building the system which effectively acts under such beliefs – they offer promising new directions on otherwise tricky problems.

3.1.3 The Difficulty of Formalizing Self-Referential Claims

However, despite the fact that self-referential claims are natural and common in everyday experience, they are difficult to incorporate into the standard Bayesian framework underlying much of AI. The core reason is that a self-referential claim allows an agent to obtain information about the world from their planned action before that action is executed. For instance, if I think that I am unlikely to successfully make money off of the stock market, then if I think of a strategy for how to invest, I must think that it is unlikely to work, so the market must work differently than I originally thought. I can then go about trying to understand how my model must have been wrong without needing to try the investment strategy, perhaps positing the existence of other traders who are already pursuing such a strategy. This ability to take your intended actions as evidence about the world, which is central to the strength of self-referential claims, is also at the center of a notorious disagreement in decision theory between evidential decision theory(EDT) and causal decision theory(CDT) [53].

In fact, traditionally this disagreement between CDT and EDT is only seen in unusual edge cases where you are playing a game with an agent which can run a perfect simulation of you such as Newcomb's problem or are in a game with your perfect twin like the twin prisoner's dilemma. This is because for there to exist a correlation between the action and some parameter of the world that is not due to the causal consequences of that action, one would seem to require a causal parent to the action, such as the source code of the agent, which would cause the otherwise unexplained correlation between the agent's action and the worlds.

However, self-referential claims assert the existence of a correlation between the agent's strategy and the world even in domains where there is no causal parent. For instance, when I tell my friend they are unlikely to be able to fix their engine, I am not asserting the existence of any pre-defined correlation between my friend and the engine design. Instead, by asserting a consequence of both my friend's strategy and the engine design, I form what in the causal inference literature is called a "collider" – a variable that is causally descendant from two or more other variables. The important feature of a collider in our context is that, when conditioning on a collider, two uncorrelated parents can become conditionally correlated. In our case, while my friend's strategy and the engine are uncorrelated, through conditioning on the self-referential claim, which is a collider, the two can become conditionally correlated. Thus, with the introduction of self-referential claims, an agent's proposed strategy can have evidence about the world without needing to posit a prior correlation. We demonstrate this effect in Section 3.2.2as we make this engine example rigorous.

This fact, that self-referential claims for colliders correlate the agent's strategy with the world, even when no prior correlations exist, greatly extends the scope of the contentions between CDT and EDT. Allowing for the possibility that this difference becomes pivotal even in banal everyday situations like fixing a car's engine.

In Section 3.2.1 we explore the approaches to self-referential beliefs which are analogous to CDT and EDT, called strategy referential and agent referential claims respectively. Strategy referential claims interpret the self-referential claim as referring to the strategy, not the agent. Thus if the agent considers taking a different strategy the self-referential claim may no longer apply. An agent-referential claim does the opposite, interprets the self-referential claim as referring to the agent rather than the agent's strategy. Thus the agent should think that if they were to change strategy they should imagine the self-referential claim would still apply.

Though these approaches make the same claims about what actually occurs, they disagree about what counterfactually occurs, and as a result, prescribe different actions. Thus, in general, these differences can be important. However, we prove in Section 3.4 that this difference between agent referential and strategy referential beliefs is often unimportant, proving in Theorem 5 that under certain assumptions the agent referential interpretation produces a strategy consistent with the strategy referential interpretation. This insight allows progress on self-referential claims to be made even without resolving all of the philosophical nuances.

Along the way, we must discuss several nuances in the behavior of strategy referential claims, caused by the circular nature of their definition. In Section 3.3, we show that, despite the circularity, strategy referential claims can be made consistent, proved formally in Theorem 4. In the remainder of the section, we discuss the nuances of strategy referential claims, giving examples of each.

3.2 Model of Decision Making

Formally, we will model an agent as operating in a decision context described as a tuple, $\langle \Theta, A, U \rangle$, where Θ is a set of unknown parameters, A is a set of possible actions, and $U : \Theta \times A \to \mathbb{R}$ is a utility function. An agent can choose between the set of possible random strategies $\Sigma \in \Delta(A)$, where $\Delta(X)$ refers to the set of probability distributions over the set X. We will refer to a distribution over the parameters $\tilde{W} \in \Delta(\Theta)$ as a *world*. For simplicity, we will assume that A and Θ are finite sets. We will extend the definition of the utility function to also apply to strategies and worlds in the natural way as follows:

$$U(\sigma, \tilde{w}) = \mathop{\mathbb{E}}_{\substack{a\sim\sigma \\ \theta\sim\tilde{w}}}[U(a, \theta)]$$

To model the beliefs of the agent, it would be most typical to simply have a distribution over worlds. Such a representation of beliefs implicitly assumes the chosen strategy of the agent and the parameters of the world are independent. However, since self-referential beliefs allow the agent to use their own choice of strategy as information about the world, this assumption may be violated. To better understand these effects, we will prefer a model where such assumptions can be made explicit.

Thus, we will intuitively be thinking of the beliefs of an agent as a joint distribution over strategies and worlds taken together, $\tilde{B} = \Delta(\tilde{W} \times \Sigma)$. As the two can be correlated, this allows the strategy to contain information about the world so that conditioning on the distinct strategies that the agent could choose could result in distinct distributions over worlds. To the issue of conditioning on zero-measure events when considering strategies that are never chosen, we will define $\tilde{B}$ through it's conditional distributions for each strategy expressed as a set of functions: $\tilde{B} : \Sigma \to \tilde{W}$. We will assume that such conditional distributions are consistent with $\tilde{B}$. More formally, for all strategies σ in the support of $\tilde{B}$ which have non-zero probability and all $\tilde{w}$ we have $\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma) = \mathbb{P}_{\tilde{B}(\sigma)}(\tilde{w})$. As such, we will abuse notation and use

$\mathbb{P}_{\tilde{B}}$ for both from here on out as there is no risk of confusion. We can then update on any event of the joint strategy-world distribution, $X \subseteq \tilde{W} \times \Sigma$ in the usual way:

$$\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma, X) = \frac{\mathbb{P}_{\tilde{B}}(X|\tilde{w}, \sigma)\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{\mathbb{P}_{\tilde{B}}(X|\sigma)}.$$

We will define the best-response set of a particular belief to be the set of strategies which, after updating on the chosen strategy as evidence, produces the distribution of worlds which maximizes the expected utility. That is,

$$BR(\tilde{B}) = \underset{\sigma \in \Sigma}{\operatorname{argmax}}\{U(\sigma, \tilde{B}(\sigma))\}.$$

We will use *opt* : $\tilde{B} \to \sigma$ to refer to the particular strategy that an agent chooses given each belief. We can then say that *opt* represents an optimal EDT agent if it always chooses a strategy which is in the best response set, $opt(\tilde{B}) \in BR(\tilde{B})$.

Though it is not the typical way of conceptualizing a CDT agent, we can express a CDT agent in this framework as one which only thinks of their strategies as evidence of some parameter of the world θ to the extent that a causal intervention on σ would effect θ. Thus, in our terminology, a CDT agent is an optimal EDT agent, but with unusual beliefs – which is why we say an *optimal* EDT agent rather than a *rational* one which one may take to imply that the beliefs are also in some sense correct or accurate.

3.2.1 Defining Self-Referential Claims

A self-referential factual claim is a claim that depends both on the strategy and on the world, and is thus defined by the subset of the two in which that claim would hold $F \subseteq \Sigma \times \tilde{W}$. For instance, "you will be unable to fix the Engine" is a claim about how the agent's chosen strategy will perform in the world.

The first important nuance to note about this claim is that it is a claim about the behavior of the agent even after the agent has updated on this claim itself. That is, it is not some tricky statement as often found in riddles of the form "you would not have be able to fix the Engine before I said this", which in those riddles often would result in the agent then being able to fix the Engine. It is the more common statement that, even after updating on this statement, the statement will hold.

Though this is a more common usage, it is in some ways more tricky formally. That is because, it is now truly self-referential in that it makes a claim about the results of the reasoning process happening for the current decision with the current beliefs. Thus it is also, in some sense, apparently circular, as the agent's belief (model of the Engine) if used to deduce a strategy (an attempt to fix the engine), which is used to interpret the self-referential claim (that particular attempt would not succeed), which then must be used to update the original belief. We will be careful to deal with this circularity explicitly and show that this can usually be resolved consistently.

	$\theta = C$	$\theta = D$
$a = C$	2	0
$a = D$	3	1

Figure 3.1: The payoff matrix of the twin prisoner's dilemma

The next important nuance to note is that, though such claims have a clear interpretation *factually*, being unambiguous about what it claims will actually occur (in this case that the agent will not be able to fix the engine themselves), it can be apparently *counterfactually ambiguous*. That is, if the optimization algorithm were, counterfactually, to choose a different strategy σ' rather than its factual strategy σ, it is unclear if the self-referential claim should still be considered true of the factual strategy or should now be consider true of the counterfactual one. Or formally, is $(\sigma, \tilde{w}) \in F$ or is $(\sigma', \tilde{w}) \in F$? Our two interpretations of self referential claims correspond to these two cases, strategy referential claims interpreting the claim as referring to the factual strategy and agent referential claims interpreting the claim as referring to the agent and thus referring to the counterfactual strategy.

As a concrete example of this distincition, consider the prisoner's dilemma, show in Figure 3.1. In this setting, we consider the traditional prisoner's dilemma where the agent has actions to either cooperate or defect, $A = \{C, D\}$, and the unknown parameters of the environment are whether the other agent cooperates or defects $\Theta = \{C, D\}$. The utility function is such that the agent receives 1 additional utility for defecting, and looses two utility of the other agent defects, shown fully in Figure 3.1.

Consider the claim "your opponent will play the same strategy as you". Formally written as, $F_{same} = \{(\sigma, \tilde{w}) | \sigma = \tilde{w}\}$. While this statement is factually unambiguous, it is counterfactually ambiguous, as it is unclear what would happen counterfactually if the agent were to take a different strategy. Concretely, suppose the agent, in reality, defects. Then it is clear that the opponent would, in reality, also defect. However, if the agent were to consider cooperating, how should they believe the opponent would behave? In the rest of this section we define the two interpretations, which are both reasonable in their own circumstances, so we can compare them formally.

3.2.1.1 The Agent Referential Interpretation

In one interpretation of F, the agent could believe that if they had changed their strategy to cooperate their opponent would also cooperate under the logic that the opponent would still choose the same strategy. This could be a reasonable interpretation in the case when the opponent is actually an exact copy of the agent, making decisions using the same program. The self referential claim could be referring to the *agent itself*, which we call the *agent referential interpretation* of F.

More formally, the agent could reason that if the agent were to not choose its usual strategy $opt(\tilde{B}')$, in this case"C", and instead choose a different strategy σ', in this case

"D", the self-referential claim would then apply to σ', and would no longer necessarily apply to $opt(\tilde{B}')$, thus concluding that the opponent would also choose "D". The agent referential interpretation of F is defined as $F^A = \{(\sigma', \tilde{w}) | (\sigma', \tilde{w}) \in F\} \subseteq \Sigma \times \tilde{W}$. Note how this update is now independent of $opt(\tilde{B}')$ depending only on the counterfactual strategy σ'.

Using the definition of Bayesian updating from the previous section we can update on the agent-referential interpretation of the self-referential beliefs as expected:

$$\mathbb{P}_{\tilde{B}'}(\tilde{w}|\sigma) = \frac{\mathbb{P}_{\tilde{B}}(F^A|\tilde{w}, \sigma)\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{\mathbb{P}_{\tilde{B}}(F^A|\sigma)} \tag{3.1}$$

$$= \frac{\mathbb{P}_{\tilde{B}}(F(\sigma, \tilde{w})|\tilde{w}, \sigma)\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{\mathbb{P}_{\tilde{B}}(F(\sigma, \tilde{w})|\sigma)}. \tag{3.2}$$

In our prisoner's dilemma example, the only consistent posterior for the Bayesian update of our self-referential belief is one which $\tilde{B}'(\sigma)(\theta) = \{\sigma : 100\%\}$ as the agent referential interpretation of F_{same} uniquely determines the opponent's strategy for each of our own counterfactual strategies. Written as $F^A_{same} = \{(\sigma', \tilde{w}) | \sigma' = \tilde{w}\} \subseteq \Sigma \times \tilde{W}$. Thus the agent with such a posterior is solving the following:

$$opt(\tilde{B}') \in \underset{\sigma \in \Sigma}{\operatorname{argmax}}\{U(\sigma, \tilde{B}(\sigma))\} \tag{3.3}$$

$$= \underset{\sigma \in \Sigma}{\operatorname{argmax}}\{U(\sigma, \sigma))\} \tag{3.4}$$

$$= \underset{\sigma \in \Sigma}{\operatorname{argmax}}\{\mathbb{P}_\sigma(a = D) - 2\mathbb{P}_\sigma(a = D)\}. \tag{3.5}$$

Thus the only optimal strategy under the agent-referential interpretation of F_{same} is to cooperate deterministically.

Note also, that the agent referential interpretation of F is not circular. Since the beliefs about the counterfactual are determined separately from the factually chosen strategy the factual choice of strategy itself does not matter for forming the beliefs, and thus their is no circularity – the beliefs determine the strategy as usual without influence the other direction. This lack of circularity allows the agent-referential interpretation to avoid the many nuances that come from circularity which our other interpretation will suffer from and which we discuss at length in Section 3.3.

3.2.1.2 The Strategy Referential Interpretation

An alternative interpretation of F is that they could instead believe that if the agent were to deviate from the factual strategy $opt(\tilde{B}') = D$ the opponent would still defect, under the logic that they play the strategy that the agent does factually. This could be a reasonable interpretation in the case where the opponent is, in fact, just a pre-defined strategy written on a chalk board, which long ago was generated arbitrarily, for instance by the flip of a coin. In this case being told that the chalk board happened to display your strategy, would give no reason that it would still match your strategy if that strategy were to change. In such cases the self-referential claim is referring just to the *agent's strategy* rather than the agent itself, thus we call such an interpretation the *strategy referential interpretation* of F.

More formally, if the agent were to consider deviating from the usual strategy $\mathit{opt}(\tilde{B}')$, in this case "D", and instead choosing a different strategy, σ', in this case "C", the self-referential claim would not necessarily apply to the new strategy σ', but would instead still apply to $\mathit{opt}(\tilde{B}')$, in this case implying the opponent would still choose "D". The strategy referential interpretation of some claim F can be defined formally as $F^S = \{(\sigma', \tilde{w}) | (\mathit{opt}(\tilde{B}'), \tilde{w}) \in F\} \subseteq \Sigma \times \tilde{W}$. In this case $F^S_{same} = \{(\sigma, \tilde{w}) | \mathit{opt}(\tilde{B}') = \tilde{w}\}$. Note how the conditioning is independent of the counterfactual strategy σ' depending only on the factual strategy $\mathit{opt}(\tilde{B}')$.

Using the definition of Bayesian updating from the previous section we can update on the strategy-referential interpretation of the self-referential beliefs as expected:

$$\mathbb{P}_{\tilde{B}'}(\tilde{w}|\sigma) = \frac{\mathbb{P}_{\tilde{B}}(F^S|\tilde{w}, \sigma)\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{\mathbb{P}_{\tilde{B}}(F^S|\sigma)} \tag{3.6}$$

$$= \frac{\mathbb{P}_{\tilde{B}}(F(\mathit{opt}(\tilde{B}'), \tilde{w})|\tilde{w}, \sigma)\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{\mathbb{P}_{\tilde{B}}(F(\mathit{opt}(\tilde{B}'), \tilde{w})|\sigma)}. \tag{3.7}$$

Note that, unlike in the case of the agent-referential interpretation, in the case of the strategy referential claims, the definitions are circular. The belief $\tilde{B}'$ is used to define the optimal strategy $\mathit{opt}(\tilde{B}')$ which is, in turn, used to define $\tilde{B}$ itself. Thus, unlike the agent-referential interpretation, we need to be careful with circularity when talking about strategy referential interpretation.

In this case, the circularity works out unambiguously. In our prisoner's dilemma example, taking $\mathit{opt}(\tilde{B}')$ to be arbitrary, the only consistent posterior for the Bayesian update of our self-referential belief is one which $\tilde{B}'(\sigma)(\theta) = \{\mathit{opt}(\tilde{B}') : 100\%\}$ as the agent referential interpretation of F uniquely determines the opponent's strategy given the factually chosen strategy. Thus the agent with such a posterior is solving the following:

$$\mathit{opt}(\tilde{B}) \in \underset{\sigma \in \Sigma}{\operatorname{argmax}}\{U(\sigma, \tilde{B}(\sigma))\} \tag{3.8}$$

$$= \underset{\sigma \in \Sigma}{\operatorname{argmax}}\{U(\sigma, \mathit{opt}(\tilde{B}')))\} \tag{3.9}$$

$$= \underset{\sigma \in \Sigma}{\operatorname{argmax}}\{\mathbb{P}_\sigma(a = D) - 2\mathbb{P}_{\mathit{opt}(\tilde{B}')}(a = D)\}. \tag{3.10}$$

Since the second term is independent of σ, the only optimal strategy is to defect deterministically, so $\mathit{opt}(\tilde{B}') = \{D : 100\%\}$. The pair $\tilde{B}'(\sigma)(\theta) = \{\mathit{opt}(\tilde{B}') : 100\%\}$ and $\mathit{opt}(\tilde{B}') = \{D : 100\%\}$ are thus the only consistent pair for the strategy referential interpretation of the self-referential claim in question. While the circularity works out in this case, this circularity of strategy referential beliefs will lead to many subtleties and counterintuitive examples, discussed in Section 3.3. Dispite these nuances, we will find that, when we are sufficiently careful, the strategy referential interpretation can often be made consistent, as we prove in Theorem 4. However, the ease with which the agent-referential interpretation will bypass these nuances should serve as some reason to prefer the agent referential interpretation in practice.

3.2.1.3 The Significance of Counterfactual Ambiguity

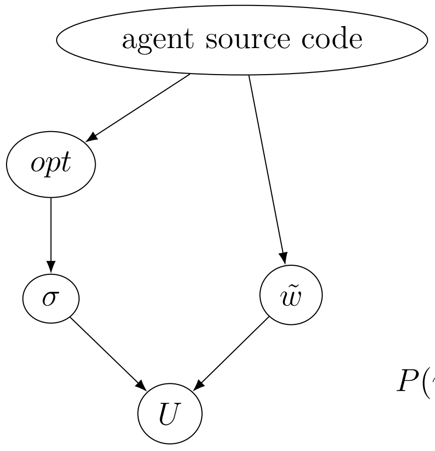
This example demonstrates that, in some settings, the counterfactual ambiguity between the agent-referential and strategy referential interpretations can be important. If we look at the factual content of these two interpretations of F, they are the same. This is because when the agent considers changing their strategy to the strategy that they usually play, the "counterfactual" and "factual" strategies coincide, $opt(\tilde{B}) = \sigma'$ and thus the two interpretations make the same prediction. That is, both claim that the strategy of the opponent will be the same as that of the agent. They only differ their predictions of what would happen if the agents chose strategies which it, in fact, does not. As such it is easy to confuse these two types of claims as they both assert the same fact about what actually happens.

However, while these two kinds of claims are in no way distinguishable based on the observed outcome the effects on the beliefs of the agents, and thus the behavior of the policy can be dramatically different. We have shown that the agent which interprets "your opponent will play the same strategy as you" as agent-referential will prefer to cooperate, where an agent which interprets "your opponent will play the same strategy as you" as strategy referential will defect. Thus, in this case, this counterfactual ambiguity is important.

In Section 3.4, we will show that this example is a special case, and that other than this kind of special case, there is no ambiguity. Thus much of the time the strategy referential and agent referential interpretations coincide not just in their factual claims, but in the resulting behavior of the agents. This means that, in many applications, the differences between these approaches can be safely ignored. Taken together with the circularity of strategy referential causing a dramatically increased amount of care and nuance, it may be prudent to prefer the agent-referential interpretation to the strategy referential interpretation, even if one beliefs the strategy referential interpretation is in some sense correct, in settings where we know this ambiguity is unimportant.

3.2.2 Causal Models

Some of the counterintuitive and useful aspects of self-referential claims can only be understood when viewed causally. This is because the utility of self-referential claims comes from the correlation between the strategy and the world which allows the agent to use their own strategy as evidence about the world. This leads many first to consider direct correlations where either the strategy effects the world, the world effects the strategy, or they share a mutual cause – situations in which there are other agents, clones of oneself, or other special circumstances. While such cases do occur, these examples ignore a much more common and useful case. Even if the strategy and world are uncorrelated in reality, they are often still correlated conditional on the self-referential claim itself, as the claim acts as a collider. As this may initially seem counterintuitive, the rest of this section will be dedicated to formally demonstrating this effect, and showing how it allows for self referential beliefs to be used to give effective advice.



(a) The DAG of the causal model of the twin prisoner's dilemma.

$$
\begin{aligned}
P(\text{agent source code} = \text{cooperate\_bot.py}) &= \frac{1}{2}\\
P(\text{agent source code} = \text{defect\_bot.py}) &= \frac{1}{2}\\
P(\mathit{opt} = \text{C-Bot}|\text{agent source code} = \text{cooperate\_bot.py}) &= 1\\
P(\mathit{opt} = \text{D-Bot}|\text{agent source code} = \text{defect\_bot.py}) &= 1\\
P(\sigma = \{C : 100\%\}|\mathit{opt} = \text{C-Bot}) &= 1\\
P(\sigma = \{D : 100\%\}|\mathit{opt} = \text{D-Bot}) &= 1\\
P(\tilde{w} = \{C : 100\%\}|\text{agent source code} = \text{cooperate\_bot.py}) &= 1\\
P(\tilde{w} = \{D : 100\%\}|\text{agent source code} = \text{defect\_bot.py}) &= 1\\
U(\sigma, \tilde{w}) = P(\sigma = D) - 2 * P(\tilde{w} = D)
\end{aligned}
$$

(b) The probabilities in the twins prisoner's dilemma.

Figure 3.2: A formal description of the twin prisoner's dilemma as a causal model.

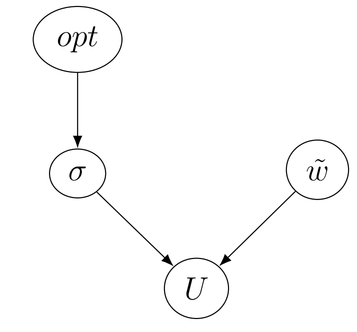
To begin, a causal Bayesian network [52] will be defined as a tuple $\langle V, E, P, \mathbf{P}^* \rangle$ such that V represents a set of variables thought of as nodes, E represents a set of directed edges between those variables, P represents the probability distribution over variables without intervention and $\mathbf{P}^*$ represents the set of possible probability distributions after interventions on variables in V. Such that the following three conditions hold for every intervention on a subset $X \subseteq V$.

- P_x is Markov relative to G, that is $P(x_1, \ldots, x_n) = \prod_i P(x_i|pa_i)$ for pa_i being the parents of x_i in G.
- $P_x(v) = 1$ for all $v \in X$ where v is consistent with x.
- $P_x(v|pa_i) = P(v|pa_i)$ for all $v \notin X$ whenever pa_i is consistent with $X = x$.

When using causal Bayesian networks to model the beliefs of an agent in a particular decision context, we will assume that the variables V, contain $\Theta, \Sigma, \mathit{opt}$ and the utility in $\mathbb{R}$.

For instance, we can formalize the example discussed in the last section, shown in Figure 3.2 and Figure 3.3. We show two plausible formulations of the problem, corresponding to the case where the opponent is an exact copy of the agent, and one corresponding to the case where the opponent is a strategy written on a chalkboard which may happen to be the same as the strategy chosen by the agent.

In the first case, where the opponent is an exact copy of the agent, there is a parent node of both the agent's strategy optimization algorithm *opt* and the parameters of the world $\tilde{w}$, which is a shared specification of the agent itself agent.py. The optimization algorithm



(a) The DAG of the causal model of the twin prisoner's dilemma, with the chalkboard interpretation

$$\begin{aligned}
P(\mathit{opt} = \text{D-Bot}) &= 1\\
P(\sigma = \{C : 100\%\} | \mathit{opt} = \text{C-Bot}) &= 1\\
P(\sigma = \{D : 100\%\} | \mathit{opt} = \text{D-Bot}) &= 1\\
P(\tilde{w} = \{D : 100\%\}) &= 1\\
U(\sigma, \tilde{w}) &= P(\sigma = D) - 2 * P(\tilde{w} = D)
\end{aligned}$$

(b) The probabilities in the chalkboard interpretation

Figure 3.3: A formal causal model describing the prisoner's dilemma with the chalkboard interpretation.

opt chooses the strategy σ and the variables σ and $\tilde{w}$ are both parents of the utility which can be computed as usual. This causal model can be shown to match the agent-referential interpretation of F in our previous section. That is, we can produce the beliefs $\tilde{B}(\tilde{w}|\sigma, F)$ by conditioning on the strategy σ in this causal model:

$$\mathbb{P}_{\tilde{B}}(\tilde{w} = \{C : \%100\} | \sigma = \{C : \%100\}) = 1 \tag{3.11}$$
$$\mathbb{P}_{\tilde{B}}(\tilde{w} = \{D : \%100\} | \sigma = \{D : \%100\}) = 1 \tag{3.12}$$

In the second case, where the opponent is a strategy written on a chalkboard which happens to be the same strategy chosen by the agent, the causal model is different, shown in Figure 3.3. There is no longer a shared parent between *opt* and $\tilde{w}$, instead $\tilde{w}$ is the deterministic strategy $\{D : 100\%\}$. This causal model can be shown to match the strategy-referential interpretation of F in our previous section. That is, we can produce the beliefs $\tilde{B}(\tilde{w}|\sigma)$ by conditioning on the strategy σ in this causal model:

$$\mathbb{P}_{\tilde{B}}(\tilde{w} = \{D : \%100\} | \sigma = \{C : \%100\}) = 1 \tag{3.13}$$
$$\mathbb{P}_{\tilde{B}}(\tilde{w} = \{D : \%100\} | \sigma = \{D : \%100\}) = 1 \tag{3.14}$$

At this point a reader maybe tempted to jump to some conclusion about the relationship between correlation, causation, and whether the self-referential belief involved should be interpreted as agent-referential or strategy-referential. However, such conclusions are not as easy to draw as this example suggests. For instance, the example called simply, *Banana* in Hitchcock [28] has the same causal model as the twin's prisoner's dilemma, but ought not be interpreted in agent referential terms. Thus the correct normative interpretation can quite subtle, and so we will defer such discussions to future work. For simplicity we will assume that we are working with the causal model *effectively believes* such that the agent acts as-if it is performing evidential decision theory within it.

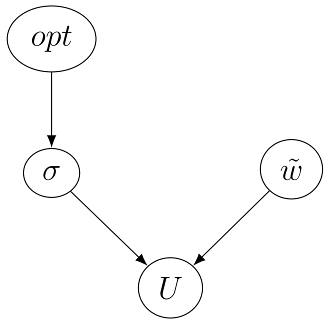
For the remainder of this work we will set aside these instances in which there are ancestors of the strategy or optimization algorithm in order to focus on less exotic cases. In such cases it is often thought that this degree of nuance can be avoided since EDT and CDT largely agree. From the standpoint of self-referential claims, which require a correlation between the strategy and the world, it could be thought that there is not much to be said, as one could believe such a correlation would apparently require a common ancestor of the strategy to serve as a mutual cause with the world. We focus on this case specifically to correct this common misconception.

In the next section we will show that, even when there are no causal parents to the strategy and thus no correlation between the strategy and the world in reality, such a correlation can exist conditional on the specification due the self-referential claim serving as collider, making the strategy and the world correlated conditional on that self referential claim. For instance, while my friend would not obviously correlate with the engine they are attempting to fix in any obvious way, if I tell my friend they will be unable to fix the engine, my friend's planed attempts to fix the engine now have information about the engine – he knows that they will not work. Thus once an agent receives advice about the effects of their own strategy the agent is unavoidably in a situation where their strategy is correlated with the world in a way that is not captured by the interventions on their strategy.

This insight extends the domain of self-referential claims from the exotic edge cases involving clones, to the the practical every day cases such as advising your friend that they will be unable to fix their engine. At the same time, this insight apparently extends the differences between strategy referential and agent referential interpretations, which mirror the differences between CDT and EDT, to everyday practical situations. Now correlation between your strategy and the world does not require an exotic scenario, but only requires that you take some prediction of the effects of your behavior seriously. Luckily, we will show with Theorem 4 and Theorem 5 that choice between strategy referential and agent referential beliefs is often still a distinction without a difference. Thus progress can be made in a wide variety of practical cases without settling every tricky philosophical case.

3.2.3 Self referential claims without prior correlation

A common misconception is that for your strategy to contain information about the world, it must have a prior correlation with the world. While this is true in a vacuum, once one has conditioned on the self referential claim itself, there could be correlation between the strategy and the world where no prior correlation existed. In this section we will demonstrate this effect with a simple example, to develop a strong understanding of the underlying mechanics. In the next section we will show how these mechanics manifest in more complex examples, showing how self-referential beliefs can serve as a valuable communication channel for efficiently correcting a model.



(a) A causal model of the guess-the-bit collider example

$$
\begin{aligned}
P(\mathit{opt} = \text{opt-0}) &= \frac{1}{3} \\
P(\mathit{opt} = \text{opt-1}) &= \frac{1}{3} \\
P(\mathit{opt} = \text{opt-}\frac{1}{2}) &= \frac{1}{3} \\
P(\sigma = \{0 : 100\%\} | \mathit{opt} = \text{opt-0}) &= 1 \\
P(\sigma = \{1 : 100\%\} | \mathit{opt} = \text{opt-1}) &= 1 \\
P(\sigma = \{0 : 50\%, 1 : 50\%\} | \mathit{opt} = \text{opt-}\frac{1}{2}) &= 1 \\
P(\tilde{w} = \{0 : 100\%\}) &= \frac{1}{2} \\
P(\tilde{w} = \{1 : 100\%\}) &= \frac{1}{2} \\
U(\sigma, \tilde{w}) &= P(\sigma = \tilde{w})
\end{aligned}
$$

(b) The probabilities in the guess-the-bit collider example

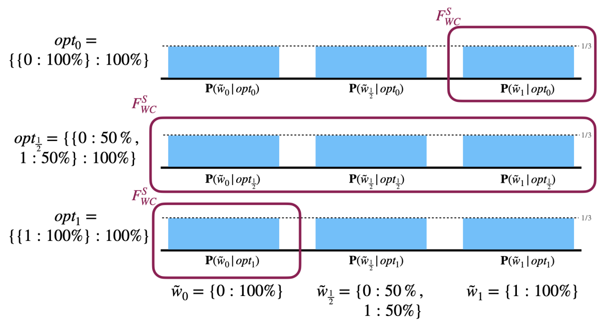
As a minimal example, take the "guess the bit" problem, show in Figure 3.4b. We can describe this problem as a simple causal model where the optimization algorithm causes the strategy strategy which causes the action, the world causes the parameters, and the action and the parameters cause the utility. In this causal model the distribution over worlds is the same independent of the strategy chosen by the agent, as shown in Figure 3.5.

We will consider the self-referential claim, "the world is one of the worst for your policy", or formally:

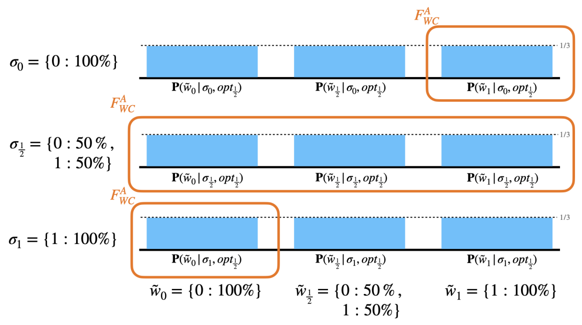
$$F_{WC} = \{(\sigma, \tilde{w}) | \tilde{w} \in \underset{\tilde{w}' \in \tilde{W}}{\operatorname{argmin}} \{U(\sigma', \tilde{w}')\}\}$$

Intuitively, if told the world is one of the worst for your policy, you would then be able to learn something about the world by looking at your policy and determining which of the worlds would make it perform the worst. We will show this intuition formally, by dealing with the strategy referential and agent referential cases separately. Note that, in both cases, this self-referential claim is conditioning on the outcome of the utility variable – in this case asserting it will be as low as possible for the particular policy. Thus, in both cases, the strategy and world may be conditionally correlated where they were initially independent.

In each case, the correlation manifests itself differently. In the case of agent-referential claim we will find that, since the claim refers to the fixed agent, the world is made to correlate with the strategy that the agent chooses. In the case of strategy-referential claim, we find the opposite. Since the claim refers to the fixed strategy for a given optimizer, the correlation will manifest between the optimizer and the world. Thus resolving uncertainty about which strategy the optimizer chooses resolves the uncertainty about what world that optimizer is



(a) The initial distribution over the *opt*, where the combinations of *opt* and $\tilde{w}$ which satisfy the strategy referential claim F^S_{WC} are outlined in purple.



(b) The initial distribution over the σ, where the combinations of σ and $\tilde{w}$ which satisfy the agent referential claim F^A_{WC} are outlined in orange.

Figure 3.5: Bar charts depicting the conditional probability distributions over worlds conditions on *opt* and σ.

in.

More concretely, in agent-referential case, the we are conditioning on the agent-referential claim $F^A_{WC} = \{(\sigma', \tilde{w}) | (\sigma', \tilde{w}) \in F\} \subseteq \Sigma \times \tilde{W}$ as defined in Section 3.2.1.1. In this case, $F^A_{WC} = \{(\sigma', \tilde{w}) | \tilde{w} \in \text{argmin}_{\tilde{w}' \in \tilde{W}} \{U(\sigma', \tilde{w}')\}\}$ That is, we are conditioning on the counterfactual strategy satisfying the self-referential claim rather than the factual strategy. We can compute the posterior belief in the usual way, note that the only consistent behavior of the optimizer is to choose as $opt(\tilde{(}B) = \text{opt-}\frac{1}{2}$, which can be used to compute the Bayesian update:

$$\mathbb{P}_{\tilde{B}'}(\tilde{w}|\sigma) = \frac{\mathbb{P}_{\tilde{B}}(F^A_{WC}|\tilde{w}, \sigma)\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{\mathbb{P}_{\tilde{B}}(F^A_{WC}|\sigma)}. \tag{3.15}$$

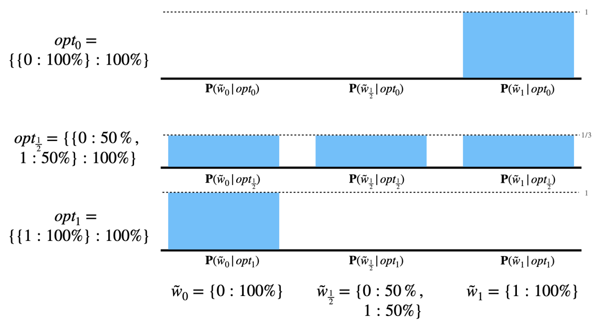
$$= \frac{\mathbb{P}_{\tilde{B}}(F_{WC}(\sigma, \tilde{w})|\tilde{w}, \sigma)\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{\mathbb{P}_{\tilde{B}}(F_{WC}(\sigma, \tilde{w})|\sigma)}. \tag{3.16}$$

By analyzing each case, it can be shown that:

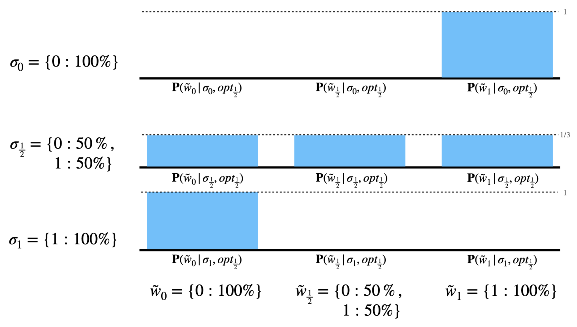
$$\mathbb{P}_{\tilde{B}'}(\tilde{w} = \{0 : 100\%\} | opt = \text{opt-1}) = 1 \tag{3.17}$$

$$\mathbb{P}_{\tilde{B}'}(\tilde{w} = \{1 : 100\%\} | opt = \text{opt-0}) = 1 \tag{3.18}$$

$$\forall \tilde{w}\ \mathbb{P}_{\tilde{B}'}(\tilde{w} | opt = \text{opt-}\frac{1}{2}) = \frac{1}{3} \tag{3.19}$$



(a) The distribution over the *opt* after conditioning on the agent referential claim F^A_{WC}



(b) The distribution over the σ after conditioning on the agent referential claim F^A_{WC}

Figure 3.6: Bar charts depicting the conditional probability distributions over worlds conditions on *opt* and σ, after conditioning on the agent referential claimF^A_{WC}.

The same relations hold for the actions even conditioned on the optimization algorithm:

$$\forall opt\ \mathbb{P}_{\tilde{B}'}(\tilde{w} = \{0 : 100\%\}|opt, \sigma = \{1 : 100\%\}) = 1 \tag{3.20}$$

$$\forall opt\ \mathbb{P}_{\tilde{B}'}(\tilde{w} = \{1 : 100\%\}|opt, \sigma = \{0 : 100\%\}) = 1 \tag{3.21}$$

$$\forall opt\ \forall \tilde{w}\ \mathbb{P}_{\tilde{B}'}(\tilde{w}|opt, \sigma = \{0 : 50\%, 1 : 50\%\}) = \frac{1}{3} \tag{3.22}$$

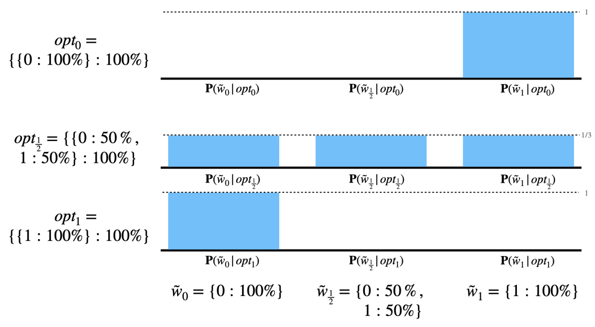
$$\tag{3.23}$$

This update can be visualized as illustrated in Figure 3.6. In this figure, it becomes obvious that, for every optimization algorithm, the initially independent strategy and world are now dependent in the way shown, due to conditioning on a shared feature.

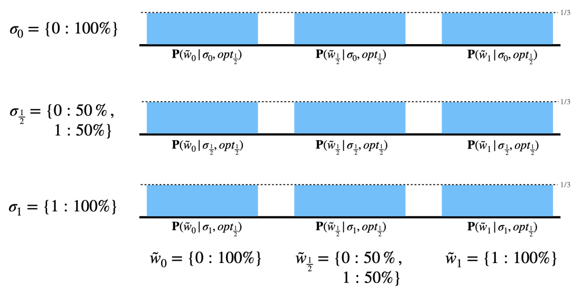
The strategy-referential case is only slightly different. We proceed by conditioning on the strategy referential claim $F^S = \{(\sigma', \tilde{w})|(opt(\tilde{B}'), \tilde{w}) \in F\} \subseteq \Sigma \times \tilde{W}$ defined in Section 3.2.1.2. In this case, $F^S_{WC} = \{(\sigma', \tilde{w})|\tilde{w} \in \operatorname{argmin}_{\tilde{w}' \in \tilde{W}}\{U(opt(\tilde{B}'), \tilde{w}')\}\}$. That is, we are conditioning on the factual strategy satisfying the self-referential claim rather than the counterfactual strategy. We can compute the posterior belief in the usual way:

$$\mathbb{P}_{\tilde{B}'}(\tilde{w}|\sigma) = \frac{\mathbb{P}_{\tilde{B}}(F^S_{WC}|\tilde{w}, \sigma)\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{\mathbb{P}_{\tilde{B}}(F^S_{WC}|\sigma)}. \tag{3.24}$$

$$= \frac{\mathbb{P}_{\tilde{B}}(F_{WC}(opt(\tilde{B}'), \tilde{w})|\tilde{w}, \sigma)\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{\mathbb{P}_{\tilde{B}}(F_{WC}(opt(\tilde{B}'), \tilde{w})|\sigma)}. \tag{3.25}$$



(a) The distribution over the *opt* after conditioning on the strategy referential claim F^S_{WC}



(b) The distribution over the σ after conditioning on the strategy referential claim F^S_{WC}

Figure 3.7: Bar charts depicting the conditional probability distributions over worlds conditions on *opt* and σ, after conditioning on the strategy referential claim F^S_{WC}.

By analyzing each case, it can be shown that:

$$\mathbb{P}_{\tilde{B}'}(\tilde{w} = \{0 : 100\%\} | opt = \text{opt-1}) = 1 \tag{3.26}$$

$$\mathbb{P}_{\tilde{B}'}(\tilde{w} = \{1 : 100\%\} | opt = \text{opt-0}) = 1 \tag{3.27}$$

$$\forall \tilde{w}\ \mathbb{P}_{\tilde{B}'}(\tilde{w} | opt = \text{opt-}\frac{1}{2}) = \frac{1}{3} \tag{3.28}$$

The same relations hold for the actions even conditioned on the optimization algorithm:

$$\forall \sigma\ \mathbb{P}_{\tilde{B}'}(\tilde{w} = \{0 : 100\%\} | opt = \text{opt-1}, \sigma) = \mathbb{P}_{\tilde{B}'}(\tilde{w} = \{0 : 100\%\} | opt = \text{opt-1}) = 1$$
$$\forall \sigma\ \mathbb{P}_{\tilde{B}'}(\tilde{w} = \{1 : 100\%\} | opt = \text{opt-0}, \sigma) = \mathbb{P}_{\tilde{B}'}(\tilde{w} = \{1 : 100\%\} | opt = \text{opt-0}) = 1$$
$$\forall \sigma\ \forall \tilde{w}\ \mathbb{P}_{\tilde{B}'}(\tilde{w} | opt = \text{opt-}\frac{1}{2}, \sigma) = \mathbb{P}_{\tilde{B}'}(\tilde{w} | opt = \text{opt-}\frac{1}{2}) = \frac{1}{3}$$

The effect of this update can be visualized as in Figure 3.7. Note that it has a slightly different effect. Conditioned on each optimization algorithm, the strategy is uncorrelated with the world, as the world only correlates with the factual output of the optimization algorithm, not the counterfactual one. However, there is a correlation between the world and the optimization algorithm itself. Thus one can get information about the world by resolving uncertainty about the optimization algorithm.

In both cases, conditioning on some aspect of the utility node forms a collider between the strategy and world. Thus the strategy and world which were initially independent are conditionally dependent. This justifies the intuition that, if you are told some property about the behavior of your strategy in the world, you can then learn something about the world by looking at your strategy. Though this correlation manifests in both the agent-referential and strategy referential cases, it manifests differently. In the agent referential case the correlation is between the counterfactual strategy and the world, and in the strategy referential case the correlation is between the factual strategy produced by the optimizer and the world.

Now that we have understood this simple case, we return to the motivational engine example, showing that self referential beliefs are necessary for effective advice. Such examples show how, despite its initially counterintuitive nature, self referential claims allow us to directly express a kind of claim, without which communication would be unacceptably inefficient.

3.2.3.1 Self Referential Claims as Effective Advice

When giving advice, it is often difficult to pinpoint exactly where the listener's model of the world is incorrect or incomplete. Even when knowing where it is incorrect or incomplete, it is often difficult to explain such misconceptions completely, as having a correct model may be the result of years of formal training and on-the-ground experience. In such cases, it is often easier to point to the important symptoms of the problem, so the listener themselves can do the work that it takes to remove that symptom.

Such is the case with our motivational engine example, your friend incorrectly thinks that they can fix the engine themself, but correcting their model would be time intensive. All you need to do to help your friend is to change one belief, that they incorrectly believe they will be successful in fixing the engine. The ability to communicate this directly, so that your friend can resolve their own misconceptions, is critical to saving the interaction from a drawn-out process where you slowly explain the workings of an engine to your stubborn friend. While this story is intuitive, the job of this section is to do the difficult work of making this formal.

Throughout this section we will use the agent-referential interpretation of "you will be unable to fix the engine", as it is easier to analyze than the strategy referential case, where one must be concerned with finding consistent fixed points. Theorem 4 shows that there is a consistent fixed point and Theorem 5 and shows that the agent-referential analysis will result in the same consistent behavior as the strategy referential analysis, though with some nuances mentioned in Section 3.3, so this preference to analyze the easier model in this case gives information about the behavior of both interpretations.

As a simplified model, suppose that the situation faced by the agent is as described in Figure 3.9 and Figure 3.9. There is only one thing wrong with the engine chosen from the set $\Theta = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$ and the agent may only take one action from the set $A = \{a_{pay}, a_1, a_2, a_3, a_4, a_5, a_6\}$. The agent gets 25 utility of the engine is fixed and 5 utility to pay for a mechanic. If the agent chooses a_{pay} then, regardless of the problem, the engine is fixed, receiving 20 utility – 25 from the fixed engine minus 2 from paying the mechanic. Alternatively, the agent can aim to fix the engine themselves, choosing one of the remaining actions $A = \{a_{pay}, a_1, a_2, a_3, a_4, a_5, a_6\}$. If the agent successfully fixes the engine they would receive a total utility of 25, having saved the money from paying the mechanic. However, if they fail to fix the engine themselves, the engine is not fixed and they receive 0 utility. Thus an optimal agent will choose a_{pay} if it has a higher expected value given their subjective prob-

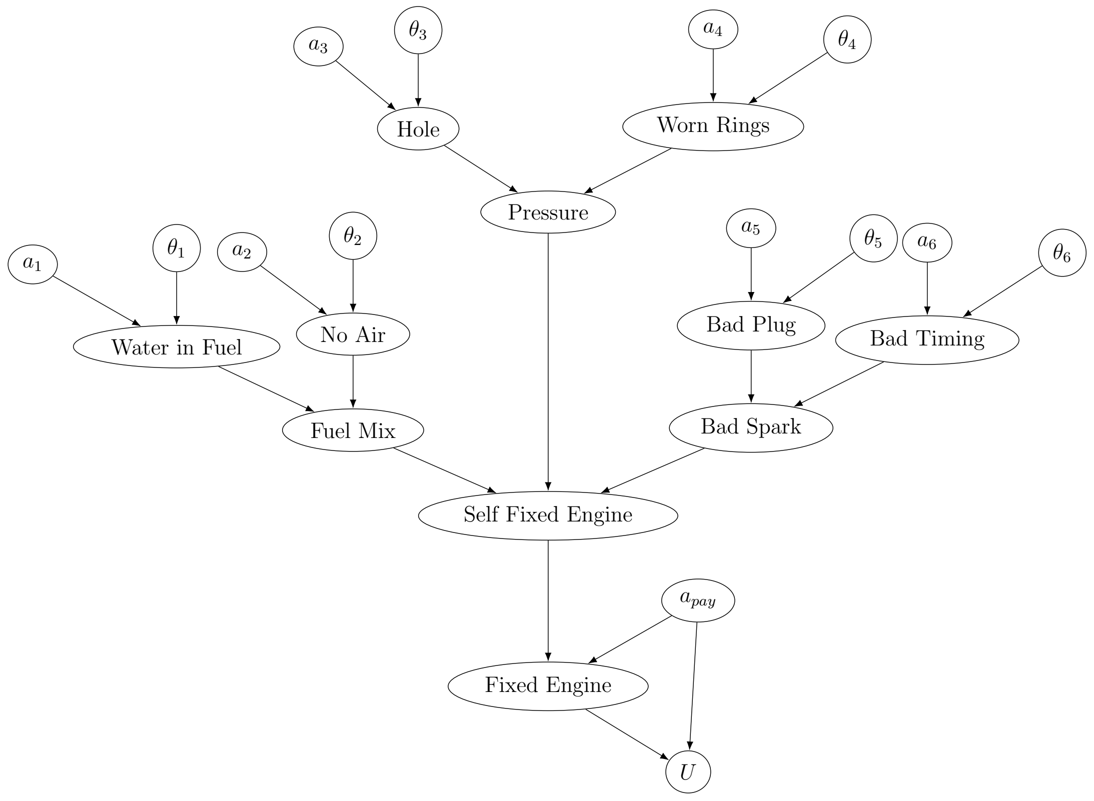


Figure 3.8: A causal model of the engine example.

$$
\begin{aligned}
U &= \mathbb{1}(\text{Fixed Engine})25 - \mathbb{1}(a_{pay})5 & \text{Water in Fuel} &= \theta_1 \lor \neg a_1 \\
\text{Fixed Engine} &= \text{Self Fixed Engine} \lor a_{pay} & \text{No Air} &= \theta_2 \lor \neg a_2 \\
\text{Self Fixed Engine} &= \text{Fuel Mix} \land \text{Pressure} \land \text{Bad Spark} & \text{Hole} &= \theta_3 \lor \neg a_3 \\
\text{Fuel Mix} &= \neg\text{Water in Fuel} \land \neg\text{No Air} & \text{Worn Rings} &= \theta_4 \lor \neg a_4 \\
\text{Pressure} &= \neg\text{Hole} \land \neg\text{Worn Rings} & \text{Bad Plug} &= \theta_5 \lor \neg a_5 \\
\text{Bad Spark} &= \neg\text{Bad Plug} \land \neg\text{Bad Timing} & \text{Bad Timing} &= \theta_6 \lor \neg a_6
\end{aligned}
$$

Figure 3.9: An explicit description of the probabilities between nodes in the DAG of the Engine example. For simplicity all internal edges of this DAG are deterministic, so we describe how to compute each node as a function of its parents. We describe the uncertainty over the parameters and actions separately as we walk through the example.

$$\begin{aligned}
&\mathbb{P}_{\tilde{B}'}(\tilde{w} = \{\theta_1 : 100\%\}) = 0.9\\
&\mathbb{P}_{\tilde{B}'}(\tilde{w} = \{\theta_2 : 100\%\}) = 0.09\\
&\mathbb{P}_{\tilde{B}'}(\tilde{w} = \{\theta_3 : 100\%\}) = 0.009\\
&\mathbb{P}_{\tilde{B}'}(\tilde{w} = \{\theta_4 : 100\%\}) = 0.0009\\
&\mathbb{P}_{\tilde{B}'}(\tilde{w} = \{\theta_5 : 100\%\}) = 0.00005\\
&\mathbb{P}_{\tilde{B}'}(\tilde{w} = \{\theta_6 : 100\%\}) = 0.00005
\end{aligned}
\qquad\qquad
\tilde{w}^T = \{\theta_6 : 100\%\}$$

Figure 3.10: On the left is the prior beliefs of the agent, before updating on the self-referential claim. We have constructed this belief as a geometric series only for the sake of simplicity of the Bayesian update. On the right is the ground-truth world, in which the self-referential claim is true about this agent. We assume that these beliefs are initially independent of the strategy σ.

ability estimate that they would successfully fix the engine. If $25\mathbb{P}_{\tilde{B}'}(\text{Self Fixed Engine}) \geq 20$ or $\mathbb{P}_{\tilde{B}'}(\text{Self Fixed Engine}) \geq \frac{20}{25} = 0.8$.

Fixing an engine yourself is a tricky task, and as an engine is made of many parts which have to operate together in an intricate fashion, where a failure in any one part can cause failure of the whole mechanism. For simplicity, we do not model all of this complexity, but we mode enough of it in order to make the dynamics of the Bayesian update clear. We will say that an agent can fix the engine themselves if they can get three components working: the engine needs to have the correct fuel mixture, maintain good pressure of that fuel, and to have a good spark. However, there can be multiple ways in which each of these components may fail. The Fuel mixture may contain water, or the air intake could be blocked. The Pressure could be lost because the piston cylinder may have a hole, or the rings around the piston inside the cylinder could be worn. The spark could fail to work appropriately if the timing system is off or if the spark plug has broken. For each of these failures we have a θ_i which corresponds to that failure being the true issues, and a corresponding a_i which corresponds to a fix for that issues. We will assume that only one issue occurs, there is only time for one fix to be attempted, that the fix always works for the identified issue, and that once that issue is fixed the system overall will work.

The agent we study is overconfident in their ability to fix the engine – they believe $\mathbb{P}_{\tilde{B}'}(\text{Self Fixed Engine}) = 0.9 > 0.8$ when it is actually much less. As such, this agent will aim to fix the engine themselves, and fail to do so, resulting in 0 utility rather than the 20 that they could otherwise guarantee. Formally, we describe the initial beliefs of this agent in Figure 3.10. For the simplicity of computing the Bayesian update, we construct this belief around a geometric series. The agent is 90% sure that they know the flaw of the engine is that there is water in the fuel. If they were told that there was no water in the fuel, then they realize the fuel mix could also be off if the air-intake were blocked and be 90% sure. If they were then told that they Fuel mix actually works, they would then realize that maintaining pressure is also important, first being confident that there would be a hole, then confident that there is a worn ring being 90% in both cases. Finally, if convinced that all of those are not the issue, the agent would realize that it could be a bad spark plug, but would now be uncertain as to the root cause, putting a 50% probability on each of the causes.

If aiming to fix the beliefs of this agent, we could inform them that each of these plans would not work in turn. That is, we could first inform the agent that there is not likely water in the fuel, then that it is not likely that there is a clog in the airline, and continue this process until they arrive at a suitably uncertain posterior. However, all of these updates have the same root symptom, they cause the agent to be overconfident in their ability to fix the engine. Moreover, if that belief was fixed, the agent would be near optimal and no other beliefs would need to be fixed.

Thus we aim to update on the self-referential claim that the agent has less than a 50% chance of succeeding at fixing the engine themselves, or formally:

$$F_{Eng} = \{(\sigma, \tilde{w}) | \mathbb{P}_{\tilde{B}'}(\text{Self Fixed Engine}|\sigma, \tilde{w}) \leq \frac{1}{2}\}.$$

Note that if, rather than using the strategy referential or agent referential interpretations, we condition on this self-referential claim as a fact about the strategy before the agent updated on this claim the agent still does not pay the mechanic. This is because the agent initially chooses a_1 and thus updating on F_{Eng} implies only that θ_1 is not the issue. The agent then beliefs that θ_2 is probably the issue with 90% confidence and changes their strategy to a_2. Thus after updating on F_{Eng} once the agent still does not believe F_{Eng}, it only believes that F_{Eng} was true before the update. One could continue this process several times, and the agent would still insist on trying to fix the engine themselves. What we actually mean when we tell them they are unlikely to fix the engine is not this pedantic "you were unlikely to fix the engine before hearing this utterance", but closer to "you are unlikely to fix the engine even after hearing this utterance". Thus one would hope to interpret the self-referential claim as iterating the update on F_{Eng} until one reaches a fixed point, as in the strategy referential interpretation of F_{Eng}. While this is the right intuition, there are subtle issues with the fixed-point interpretation in this case, which we defer to Section 3.3.1.2. For the remainder of this section we will instead look at the agent-referential interpretation of F_{Eng}, which is easier to analyze.

In the agent referential interpretation of $F^A_{Eng} = \{(\sigma', \tilde{w}) | \mathbb{P}_{\tilde{B}'}(\text{Self Fixed Engine}|\sigma', \tilde{w}) \leq \frac{1}{2}\}$, the analysis becomes straightforward. If the agent considers taking action a_1, they believe that θ_1 is unlikely: $\mathbb{P}_{\tilde{B}'}(\theta_1|a_1) \leq \frac{1}{2}$, similarly with $\mathbb{P}_{\tilde{B}'}(\theta_2|a_2) \leq \frac{1}{2}$, ect. Since the agent would, in none of these cases, fix the engine with probability $> 80\%$, the agent would decide to take a_{pay} to pay the mechanic to fix the engine, and thus receive the guaranteed 20 utility.

In essence, the use of a self-referential claim in this circumstance allows one the ability to correct the misconceptions of this agent with a single claim. Instead of aiming to track down all of the misconceptions in the agent's model for an engine, you offload this work to the agent themselves by pointing out the one important consequence of the misconceptions, allowing the agent to make the necessary updates to fix the root causes themselves. As such, a simple self referential claim allows one to avoid the long drawn-out process of fixing the agent's beliefs manually.

3.3 Nuances of Strategy Referential Claims

As the definition of strategy referential claims are circular, we aim to study when strategy referential claims are consistent in the first place. We will show that strategy referential claims are consistent under broad conditions. That is we will show in Theorem 4 that, for self referential claims satisfying a few natural properties, there is always an optimal agent which beliefs the strategy referential interpretation of that claim.

However, though strategy referential claims are broadly consistent, there are several surprising nuances in implementing such agents, which we will explore in depth through several examples. In particular, in Section 3.3.1.1 we will show that the conditions of Theorem 4 are necessary, showing examples where no agent can consistently believe the strategy referential claim if the conditions of the theorem are not satisfied. In Section 3.3.1.2 we will show that, even when a fixed point exists, it may be impossible to make a truthful self-referential claim about the agent as it may be design as to never implement the consistent fixed-point. Moreover, in Section 3.3.1.3, we show an example in which, even though the agent does implement the consistent fixed point, updating on the self-referential claim itself would not be sufficient to cause the posterior belief necessary for the fixed point, and so no truthful strategy referential claim can be made about the agent, and even updating on the strategy referential claim will not make it arrive at the consistent strategy-referential behavior. Finally, in Section 3.3.1.4 we show that even when there is a consistent strategy referential behavior it may not be unique, and that there are situations in which strategy referential claims form self-fulfilling prophecies, where two contradictory strategy referential claims can be truthfully made about a given agent, as once the agent updates on either strategy referential claim, that claim will be validated by the agent's resulting behavior.

3.3.1 Consistency of Strategy Referential Claims

To show that the strategy referential interpretation of self referential claims are broadly consistent, we introduce two natural conditions on the claims.

The first of these conditions we call unilateral consistency – ensuring that, for every policy π there is at least one world $\tilde{w}$ with which the self-referential claim F would apply, as otherwise the self-referential claim would directly make a claim about the policy π which the agent could unilaterally contradict. More formally, a self referential claim F is unilaterally consistent if, for any strategy $\sigma \in \Sigma$, there is a world $\tilde{w} \in \tilde{W}$ such that $(\tilde{w}, \sigma) \in F$. An example of an self referential claim F which is not unilaterally consistent would be the claim $F = \{(\sigma, \tilde{w} | \sigma = \{a_1 : 100\%\}\}$, that is, the claim that the agent chooses the strategy $\{a_1 : 100\%\}$. Such a claim not only makes a claim about the consequences of the agent's behavior, but about the agent's behavior itself, in a way that the agent could directly contradict. In this case, the F would not be a consistent strategy referential claim if $\{a_1 : 100\%\}$ was a dominated strategy. While it may be possible to reason effectively about such claims, doing so is beyond the scope of this work.

The second condition we require is that the self-referential claim is closed. Formally, we will say a self-referential claim is closed if F, as it is defined by a set in $\tilde{W} \times \Sigma$, is a closed set. That is, if a sequence of $(\tilde{w}_n, \sigma_n)$ pairs converges to $(\overline{\tilde{w}_n}, \overline{\sigma_n})$, and if for all n, $(\tilde{w}_n, \sigma_n) \in F$, then $(\overline{\tilde{w}_n}, \overline{\sigma_n}) \in F$. We show in Section 3.3.1.1 that, without this condition, we could not prove consistency. More importantly, this condition is quite natural, as it otherwise it implies a sort of discontinuity of the claim in terms of small changes in the distributions $\tilde{w}$ or σ.

With these two conditions, we are ready to prove our main theorem of this section, that any self referential claim which satisfies these two conditions is consistent. That is, there is an optimal agent which believes the strategy referential interpretation of the claim. The structure of this proof follows largely the structure of Nash's theorem proving the existence of Nash Equilibrium in finite n-player games [49]. In fact, Nash's theorem would follow from this one as "The opponent's play an optimal response to my strategy" is a self-referential claim, and the strategy referential interpretation of that claim would line up with that model.The crux of the proof is in constructing a set-valued function such that a fixed point of that function would have the desired property, and then applying Kakutani's fixed point theorem.

Theorem 4. *The strategy-referential interpretation of any unilaterally consistent self-referential claim with a closed graph can be consistently believed. That is, there is a pair $(\tilde{B}, \sigma)$, such that $\sigma \in BR(\tilde{B})$ and for all σ' $\mathbb{P}_{\tilde{B}}(F^S_\sigma | \sigma') = 1$.*

Proof. In this proof, we will only need to consider agents which believe only strategy referential claims, so for all σ, σ' we have $\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma) = \mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma')$. Thus for the course of this proof we will identify beliefs with a fixed distribution of worlds $\tilde{B}^S \in \Delta(\tilde{W})$, to which all of our definitions extend naturally.

Let F be a closed self-referential claim which is unilaterally consistent. Consider the map $G : \Delta(\tilde{W}) \times \Sigma \to 2^{\Delta(\tilde{W}) \times \Sigma}$, which is a set-valued map from $\Delta(\tilde{W}) \times \Sigma$ to itself, defined such that $(\tilde{B}'^S, \sigma') \in G(\tilde{B}^S, \sigma)$ if and only if $\mathbb{P}_{\tilde{B}'}(F^S_\sigma) = 1$ and $\sigma' \in BR(\tilde{B}^S)$. Then we aim to apply the Kakutani fixed-point theorem to show that G has a fixed point – a pair $(\tilde{B}^S, \sigma)$ such that $(\tilde{B}^S, \sigma) \in G(\tilde{B}^S, \sigma)$. Since the domain of G is clearly a a non-empty, compact and convex subset of $\mathbb{R}^n$, the remaining conditions to apply the Kakutani fixed-point theorem are:

- $G(\tilde{B}^S, \sigma)$ is non-empty for all $\tilde{B}^S, \sigma \in \Delta(\tilde{W}) \times \Sigma$
- $G(\tilde{B}^S, \sigma)$ is convex for all $\tilde{B}^S, \sigma \in \Delta(\tilde{W}) \times \Sigma$
- G has a closed graph

G is non-empty: To show that $G(\tilde{B}^S, \sigma)$ is non empty, let $(\tilde{B}^S, \sigma)$ be arbitrary. Then note that $BR(\tilde{B}^S)$ is non empty so there is a $\sigma' \in BR(\tilde{B}^S)$. Note also that, bu the unilatral consistency condition, there is a world $\tilde{w}'$ such that $(\tilde{w}', \sigma) \in F$. Thus, by constructing $\tilde{B}'^S$ to put certainty on world $\tilde{w}'$ we have $\mathbb{P}_{\tilde{B}'}(F^S_\sigma) = 1$. Putting those observations together, we have that $(\tilde{B}'^S, \sigma') \in G(\tilde{B}^S, \sigma)$ by definition, and thus $G(\tilde{B}^S, \sigma)$ is non-empty, as desired.

G is convex: To show that $G(\tilde{B}^S, \sigma)$ is convex first note that $BR(\tilde{B}^S)$ is convex, since any two strategies σ and σ' which are both a best response to $\tilde{w}$ must achieve the same score, and mixing between them will achieve that same score. That is, for all p we can construct $\sigma^p = p\sigma + (1-p)\sigma'$, or in other words, for all a_i $\mathbb{P}_{\sigma^p}(a_i) = p\mathbb{P}_\sigma(a_i) + (1-p)\mathbb{P}_{\sigma'}(a_i)$. Thus we have:

$$\begin{aligned}
U(\sigma^p, \tilde{B}^S) &= \mathop{\mathbb{E}}_{\substack{a\sim\sigma \\ \theta\sim\tilde{B}^S}}[U(a,\theta)] && (3.29)\\
&= p \mathop{\mathbb{E}}_{\substack{a\sim\sigma \\ \theta\sim\tilde{B}^S}}[U(a,\theta)] + (1-p) \mathop{\mathbb{E}}_{\substack{a\sim\sigma' \\ \theta\sim\tilde{B}^S}}[U(a,\theta)] && (3.30)\\
&= pU(\sigma, \tilde{B}^S) + (1-p)U(\sigma', \tilde{B}^S) && (3.31)\\
&= U(\sigma, \tilde{B}^S) && (3.32)
\end{aligned}$$

Where the last line follows since $U(\sigma, \tilde{B}^S) = U(\sigma', \tilde{B}^S)$, as they are both equally optimal. Thus σ^p achieves the same expected utility as σ, and $\sigma \in BR(\tilde{B}^S)$, $\sigma^p \in BR(\tilde{B}^S)$ as desired.

We can also show that the set of beliefs consistent with F for a fixed σ is also convex. Consider suppose we have beliefs $\tilde{B}^S$ and $\tilde{B}'^S$, then we can define $\tilde{B}^{pS} = p\tilde{B}^S + (1-p)\tilde{B}'^S$, or in other words, for all $\tilde{w}$ $\mathbb{P}_{\tilde{B}^{pS}}(\tilde{w}) = p\mathbb{P}_{\tilde{B}^S}(\tilde{w}) + (1-p)\mathbb{P}_{\tilde{B}'^S}(a_i)$. Thus since $\mathbb{P}_{\tilde{B}^S}(F^S_\sigma) = 1$ and $\mathbb{P}_{\tilde{B}'^S}(F^S_\sigma) = 1$, $\mathbb{P}_{\tilde{B}^{pS}}(\tilde{w})$, proving the convexity of the set of beliefs consistent with F^S_σ.

Finally, since $G(\tilde{B}^S, \sigma)$ is the Cartesian product of two convex sets, it is convex. To show this formally, consider two convex sets X, Y, two pairs $(x, y) \in X \times Y$ and $(x', y') \in X \times Y$, and some mixture $p \in [0,1]$. To show $(px + (1-p)x', py + (1-p)y') \in X \times Y$, it is sufficient to note that $px + (1-p)x' \in X$ and $py + (1-p)y' \in Y$.

G has a closed graph: Thus we know that $G(\tilde{B}^S, \sigma)$ is always non-empty and convex. Note that $BR(\tilde{B}^S)$ has a closed graph. Note also that since F has a closed graph, the set of $\tilde{B}^S$ which put probability 1 on F have a closed graph. Since the graph of G is the product of the graphs of $BR(\tilde{B}^S)$ and the set of $\tilde{B}^S$ which put probability 1 on F, and the product of two closed sets is closed the graph of $G(\tilde{B}^S, \sigma)$ must also be closed.

Applying Kakutani's fixed-point theorem Thus we know that G satisfies the conditions of the Kakutani fixed-point theorem and there is a tuple $(\tilde{B}^S, \sigma)$ such that $(\tilde{B}^S, \sigma) \in G(\tilde{B}^S, \sigma)$. Thus $\mathbb{P}_{\tilde{B}'}(F^S_\sigma) = 1$ and $(\tilde{B}^S, \sigma) \in BR(\tilde{B}^S)$ by the definition of G. Thus the *EDT* agent that believes $\tilde{B}^S$ which is certain of F^S_σ and responds with σ, as desired. $\square$

Thus we have shown that the strategy referential interpretation of any unilaterally consistent and closed self referential claim is consistent. This implies that, in a broad range of circumstances, we can make agents which have these strategy referential beliefs. However,

	θ_0	θ_1
a_0	1	0
a_1	0	1

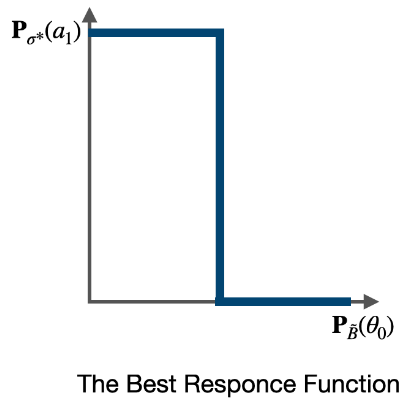


Figure 3.11: The payoff matrix and best response function for "guess the bit"

there are also several nuances in doing so. For the rest of this section we will go through examples of each of these nuances.

3.3.1.1 The Necessity of Closedness

We have already discussed how the unilateral consistency condition in necessary for the proof of Theorem 4, as otherwise one could assert that the agent chooses a dominated strategy, which is not consistent with the behavior of any optimal agent. In this section we will study what happens if we remove the closedness condition, showing that removing this condition could result in a self referential claim for which there is no consistent strategy referential interpretation.

For this we return to the "guess the bit" problem, studied previously in Section 3.2.3. This is the decision problem in which the possible environment parameter $\Theta = \{0, 1\}$ correspond to the available actions $A = \{0, 1\}$, and the agent gets 1 reward for choosing the action which matches the world, and 0 points otherwise:

$$U(a,\theta) = \begin{cases} 1 & \text{if } a = \theta \\ 0 & \text{otherwise} \end{cases}.$$

We represent this decision problem as a decision matrix in Figure 3.11. In two action, 2 parameter decision problem like "guess the bit", both the self-referential claim, the beliefs consistent with a self-referential claim, and the response of $opt(\tilde{B})$ to a particular belief can all be visualized, since the distributions over the parameters and actions are each one dimensional. Thus this setting is ideal for building intuition about the behavior of self-referential claims.

In this section we will start with a self-referential claim that is closed, and thus is consistent, and slowly remove this closedness to demonstrate exactly where the belief can no

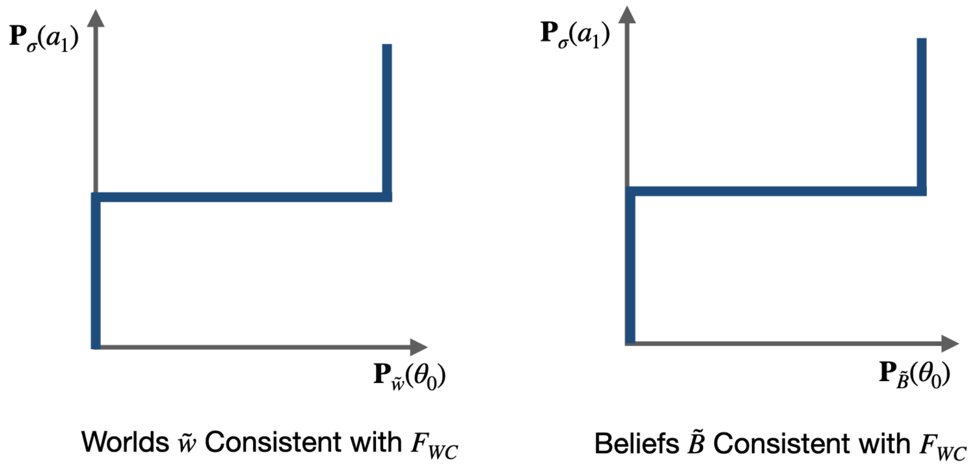


Figure 3.12: The worlds and beliefs consistent with F_{WC}.

longer be made consistent. For each case we will visualize the self-referential claim, the set of beliefs consistent with that claim, and the best-response function. The first self-referential belief to consider, which satisfies the conditions of Theorem 4 is "the world is one of the worst for your strategy", or more explicitly:

$$F_{WC} = \{(\sigma, \tilde{w}) | \tilde{w} \in \underset{\tilde{w}}{\operatorname{argmin}}\{\underset{a,\theta}{\mathbb{E}}[U(a, \theta]\}\}.$$

Any self referential claim in a 2-action 2-parameter decision problem can be visualized by plotting the probability that the true parameter is θ_0, given the probability of playing a_0, that is $\mathbb{P}_{\tilde{w}}(\theta_0)$ given $\mathbb{P}_\sigma(a_0)$. For F_{WC} this results in a continuous zig-zag across the space of joint strategy-worlds as shown in 3.12. If the strategy is puts more weight on a_0 the world is deterministically θ_1, if the strategy puts more weight on a_1 the world is deterministically a_0, and if equal weight is put on both actions all worlds are possible. The set of consistent beliefs can be visualized with the same kind of plot but instead measuring the probability by the possible beliefs $\tilde{B}$ rather than the possible worlds $\tilde{w}$. This results in the beliefs being the convex combination of all of the consistent worlds, in the case of F_{WC}, since the set of consistent worlds is already convex for each σ, the set of consistent beliefs is the same zig-sag.

Finally, the best-response function, can be visualized in a similar way shown in Figure 3.11, plotting the probability of a_0 under the optimal policy $\mathbb{P}_{\sigma^*}(a_0)$, against a belief $\tilde{B}$ with a particular probability of the true parameters being θ_0, $\mathbb{P}_{\tilde{B}}(\theta_0)$. This results in a zig-zag best-response function where beliefs in which the probability of choosing θ_0 are below 50%, the strategy chooses a_1 deterministically, beliefs in which the probability of choosing θ_0 are above 50% the strategy chooses a_0 deterministically, and the belief is that the parameters are both equally likely, all strategies are possible.

A fixed point, as described by Theorem 4,is a pair of a strategy and a belief such that the strategy is a best response to the belief and the belief is a best response to the strategy. This corresponds to a point which is in the intersection of both of these visualized sets, in this example $\sigma = \{0 : 50\%, 1 : 50\%\}$ and $\tilde{B} = \{\{0 : 50\%, 1 : 50\%\} : 100\%\}$ is in the intersection of both sets. That is, the strategy randomizes uniformly between 0 and 1, and the belief is certain that the distribution over worlds is uniformly random between 0 and 1. To check that this is a fixed point of the self-referential claim, note that the random strategy is a best response to certainty of a random environment(since everything is a best response), and that such an environment is also one of the worst for that strategy (since all worlds are equally bad for it).

To construct an example where there is, in fact, no fixed point one could try to remove the belief in probabilistic worlds. Consider the self referential claim "the world is deterministic and one of the worst for your strategy", or more explicitly:

$$F_{WCD} = \{(\sigma, \tilde{w}) | \tilde{w} \in \underset{\tilde{w}}{\operatorname{argmin}}\{\underset{a,\theta}{\mathbb{E}}[U(a, \theta] \cap \{\{\theta_0 : 100\%\}, \{\theta_1 : 100\%\}\}\}\}.$$

In this case we can again visualize the self-referential claim shown in Figure 3.13, and the corresponding beliefs. The self referential claim only changes by removing the probabilistic worlds when the strategy is uniformly random, resulting in two disconnected lines. However, the beliefs consistent with the strategy referential interpretation of this claim do not change, since an agent could believe that the world is either $\{\theta_0 : 100\%\}$ or $\{\theta_1 : 100\%\}$ but be unsure between the two with any probability. Thus the original fixed point still exists. In fact, F_{WCD} is still closed, and so Theorem 4 still applies.

To finally reach our example where no fixed point exists, we must ensure that a fixed point will still not exist once we add in all convex combinations of worlds consistent with the self-referential claim when we visualize the belief. To do this, we need to consider a more unnatural self referential claim which takes preference between $\{\theta_0 : 100\%\}$ and $\{\theta_1 : 100\%\}$ when there is a tie. Consider the self-referential claim of "the world is one of the worst for you, deterministic, and is zero if both options perform the same against your policy", or more precisely

$$F_{WCP} = \{(\sigma, \tilde{w}) | \underset{\tilde{w}}{\operatorname{argmin}}\{\underset{a,\theta}{\mathbb{E}}[U(a, \theta]\} \vee \tilde{w} \in \{\{1 : 100\%\}, \{0 : 100\%\} \vee 0 \in \underset{\tilde{w}}{\operatorname{argmin}}\{\underset{a,\theta}{\mathbb{E}}[U(a, \theta]\} \Rightarrow \tilde{w} \neq 1\}\}.$$

Though this is now a quite unnatural self-referential claim, it is now actually an example of a self-referential claim where no agent believes it's strategy-referential interpretation. If we now look at the graph of F_{WCD} in 3.14, we have simply removed a point on the bottom segment. This has the effect on the graph of the consistent beliefs no longer filling in the

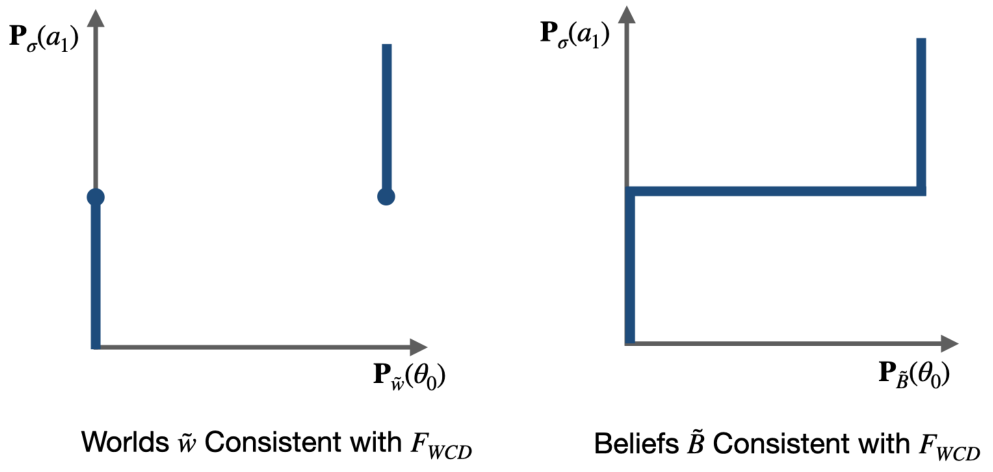

$\mathbf{P}_\sigma(a_1)$
$\mathbf{P}_{\tilde{w}}(\theta_0)$
Worlds $\tilde{w}$ Consistent with F_{WCD}
$\mathbf{P}_\sigma(a_1)$
$\mathbf{P}_{\tilde{B}}(\theta_0)$
Beliefs $\tilde{B}$ Consistent with F_{WCD}

Figure 3.13: The worlds and beliefs consistent with F_{WCD}.

gap between the two segments, as there is now only one world consistent with any strategy. Thus, it is easy to see from the images that the best-response graph and the set of consistent beliefs no longer touch, and thus have an empty intersection. An intuitive way of knowing that this self-referential claim cannot have a consistent strategy referential interpretation is that, for every strategy, the self referential belief is sufficient for the agent to know that they are sub-optimal. if the agent is playing 1 at least half the time the self-referential claim asserts that, it ought to play 0, and otherwise it ought to play 1. Since an optimal agent will never be certain that some specific action is an improvement over it's current action, and because the strategy referential interpretation of the self-referential claim asserts this is always the case, there is no way an optimal agent can consistently believe it. Note that to achieve this result, the bottom segment is no longer closed, as the strategy approaches uniform, the beliefs are in a deterministic θ_0, but the limit at the uniform strategy is in a deterministic θ_1.

This example shows that the closedness condition on F is necessary in order for the theorem statement to hold. The exposition of this example also builds some intuition as to how unnatural non-closed beliefs would be, as the issue only appears when we work to construct a somewhat unnatural counterexample. In the next few sections we will build on this visualization of strategy referential claims to demonstrate how even when closedness applies handling strategy referential claims appropriately requires appropriate nuance.

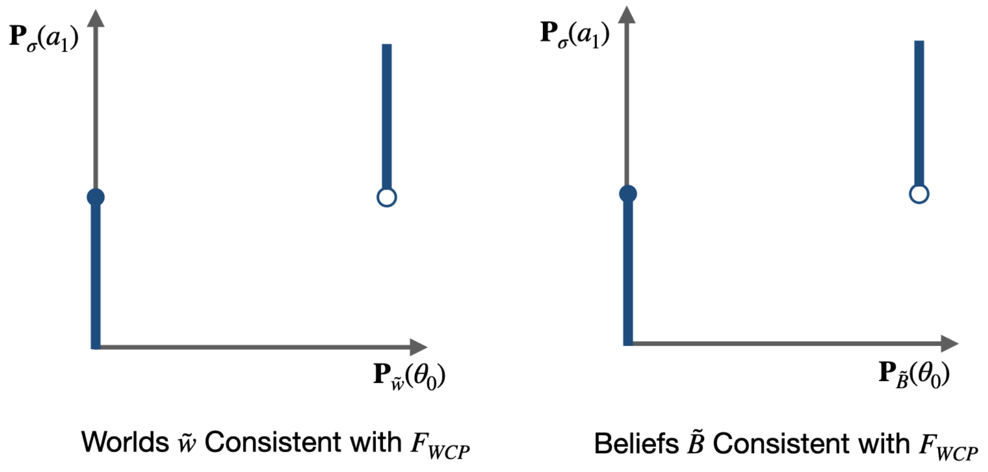


Figure 3.14: The worlds and beliefs consistent with F_{WCP}.

3.3.1.2 Stubborn Agents and Weak Equalibria

We have shown how unilaterally consistent and closed self referential claims have consistent strategy referential interpretations, and how if the claim is not unilaterally consistent or closed the strategy referential interpretation may be inconsistent. In this section we will show that even when the strategy referential interpretation is consistent, it may not be possible for a particular agent to be made to believe the claim.

For a simple example of this we can return to the "guess the bit", setting, and reconsider the self referential claim "the world is one of the worst for your strategy":

$$F_{WC} = \{(\sigma, \tilde{w}) | \tilde{w} \in \underset{\tilde{w}}{\text{argmin}} \{ \underset{a,\theta}{\mathbb{E}}[U(a, \theta]\}\}.$$

We have shown previously that this self-referential claim has a consistent strategy referential interpretation, the fixed point being when there is a uniform probability of either environment parameter being chosen $\tilde{B} = \{\{\theta_0 : 50\%, \theta_1 : 50\%\} : 100\%\}$ and the strategy chooses actions uniformly randomly $opt(\tilde{B}) = \{a_0 : 50\%, a_1 : 50\%\}$. However, notice that when an agent believes $\tilde{B}$, the uniform strategy is not strictly preferred to any other strategy. That is, in the same way as mixed Nash Equilibria, the mixed equilibria with strategy referential beliefs will be weak. As such, an optimization algorithm could be equally optimal if it chose $opt(\tilde{B}) = \{a_0 : 100\%\}$, we visualize this in Figure 3.15. Note that now *opt* can

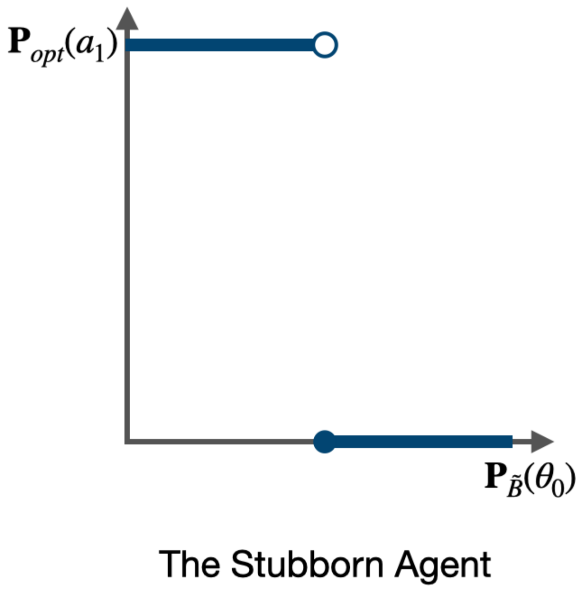


Figure 3.15: The *opt* function for a stubborn agent, which will be unable to be convinced of F^S_{WC}.

	$\theta = 0$	$\theta = 1$
a_0	1	0
a_1	0	1
a_{abs}	0.1	0.1

Figure 3.16: The payoff matrix for "guess the bit or abstain"

never believe the strategy referential claim F_{WC}, as the graph shown in Figure 3.15 no longer intersects with the possible belief graph shown in Figure 3.12, there is no fixed point.

3.3.1.3 Consistent but Unreachable Posteriors

Even if the optimization algorithm does have a fixed point, there are examples where it will not be able to reach that fixed point after a Bayesian update. For instance consider, "guess the bit or abstain", where we have added a third action a_{abs} to the original game, where you get a slight positive utility of 0.1 for deciding to abstain rather than choosing an option, as described in Figure 3.16. We will consider an agent with the prior belief for all σ $\tilde{B}(\sigma) = \{\{\theta_0 : 100\%\} : 99\%, \{\theta_0 : 50\%, \theta_1 : 50\%\} : 1\%\}$. Note that with this prior the only optimal strategy is to guess a_1 deterministically, $\sigma = \{a_1 : 100\%\}$.

We can consider the strategy referential interpretation of the self referential claim "if you guess you will most likely guess the wrong bit" or formally:

$$F_{WB} = \{(\sigma, \tilde{w}) | \mathbb{P}_{\sigma^*}((a_0 \wedge \theta_0) \vee (a_1 \wedge \theta_1) \leq 50\%\}.$$

If one believed this then they would want to abstain rather than guess the bit. It is easy to verify that this belief is closed and unilaterally consistent, so there is a fixed point. The only fixed point is to deterministically abstain $\sigma = \{a_{abs} : 100\%\}$, since any actions which guess the bit are not worth it in expectation, given the self-referential claim. Moreover, $\{\theta_0 : 50\%, \theta_1 : 50\%\}$ is a belief that supports this, which has support in the prior *tildeB*. However, note that the self-referential claim F_{WB} makes no claim about the world when the strategy is to deterministically abstain. Thus computing the strategy referential Bayesian update on the only consistent strategy from the prior belief gives no information, or to work it out formally:

$$\mathbb{P}_{\tilde{B}'}(\tilde{w}|\sigma) = \frac{\mathbb{P}_{\tilde{B}}(F^S|\tilde{w},\sigma)\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{\mathbb{P}_{\tilde{B}}(F^S|\sigma)}. \tag{3.33}$$

$$= \frac{\mathbb{P}_{\tilde{B}}(F(opt(\tilde{B}'),\tilde{w})|\tilde{w},\sigma)\mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{\mathbb{P}_{\tilde{B}}(F(opt(\tilde{B}'),\tilde{w})|\sigma)} \tag{3.34}$$

$$= \frac{1 * \mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma)}{1} = \mathbb{P}_{\tilde{B}}(\tilde{w}|\sigma) \tag{3.35}$$

Thus, the belief does not change after the Bayesian update. Since the prior belief was does not give a fixed point, and the posterior given the only fixed point in is the same as the prior, the posterior must also not give a fixed point. Thus, thought there is a consistent strategy referential claim, and though this agent would choose the fixed point strategy if it had the fixed point belief, the strategy referential update does not lead to the fixed point belief and thus does not lead to the fixed point strategy.

3.3.1.4 Self Fulfilling Prophecies

A finally surprising nuance is that strategy referential beliefs also allow for self-fulfilling prophecies. That is, two contradictory strategy-referential self-referential claims which cannot ever be true at once, but either of which would be true once the agent conditions on it being true.

To see an example of a self-fulfilling prophecy, consider the decision problem "invest or safety", in which there are two possible market conditions $\theta \in \{\text{good market}, \text{bad market}\}$ and two possible investment actions $a \in \{\text{safe}, \text{risky}\}$. The payoff matrix, as shown in Figure 3.17 gives 1 utility if the safe action is played, 2 utility if the risky action is played in a good market, and 0 otherwise.

Consider the prior over market conditions as follows $\tilde{B} = \{\frac{1}{3} : \{100\% : \text{good market}\}, \frac{1}{3} : \{100\% : \text{bad market}\}, \frac{1}{3} : \{50\% : \text{good market}, 50\% : \text{bad market}\}\}$. That is, there is a $\frac{1}{3}$ subjective probability the market is deterministically good, a $\frac{1}{3}$ subjective probability the market is deterministically bad, and a $\frac{1}{3}$ chance that the market is fundamentally uncertain

	$\theta =$ good market	$\theta =$ bad market
$a =$ invest	2	0
$a =$ safety	1	1

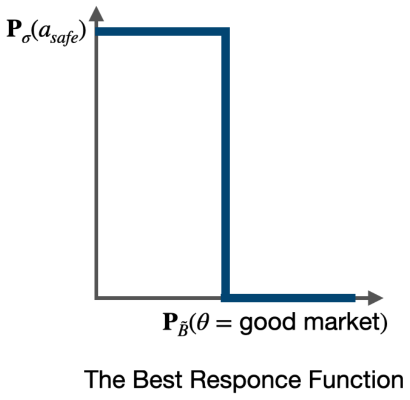


Figure 3.17: The payoff matrix and best response function for “invest or safety”

and will be good with probability $\frac{1}{2}$. Note that, with this prior, both options are equally valuable, but if any more probability is shifted towards $\theta =$ good market then $a = risky$ is strictly dominate and if any more probability is shifted towards $\theta =$ bad market then $a = safe$ is strictly dominate.

Now consider the following two self-referential claims “it is more often that “risky” pays-out than it is that “safe” defends against losses”, or the opposite “it is less often that “risky” pays-out than it is that “safe” defends against losses”. To be completely concrete, these self-referential claims can be expressed formally as the following:

$$F_{risky\_pays} = \{(\tilde{w}, \sigma) | \underset{\sigma,\tilde{w}}{\mathbb{P}}(\theta = \text{good market} \vee a = \text{risky}) > \underset{\sigma,\tilde{w}}{\mathbb{P}}(\theta = \text{bad market} \vee a = \text{safe})\}$$

$$F_{safe\_pays} = \{(\tilde{w}, \sigma) | \underset{\sigma,\tilde{w}}{\mathbb{P}}(\theta = \text{good market} \vee a = \text{risky}) < \underset{\sigma,\tilde{w}}{\mathbb{P}}(\theta = \text{bad market} \vee a = \text{safe})\}.$$

As before, we can visualize these self-referential claims, shown in Figure 3.18. Clearly these are two contradictory claims, which cannot be simultaneously true.

However, suppose that the real world was $\tilde{w} = \{50\% : \text{good market}, 50\% : \text{bad market}\}$, that is, the market condition is fundamentally random. Then it is in fact the case that, despite these claims directly contradicting each other, either of these can be claims can be truthfully made about the agent, since once the agent updates on the claim it will be true.

More explicitly, an agent with an initial belief $\tilde{B}$ as described above will, after updating on the strategy referential interpretation of F^{risky_pays}, play the risky strategy since this will remove the possibility that the market is deterministically bad. This will then ensure that $\mathbb{P}_{\sigma,\tilde{w}}(\theta = \text{good market} \vee a = \text{risky}) > \mathbb{P}_{\sigma,\tilde{w}}(\theta = \text{bad market} \vee a = \text{safe})$, making F_{risky_pays} true. Symmetrically, the opposite could happen – that same agent will, after updating on the strategy referential interpretation of F_{safe_pays}, play the safe strategy since this will remove the possibility that the market is deterministically good. This will then ensure that

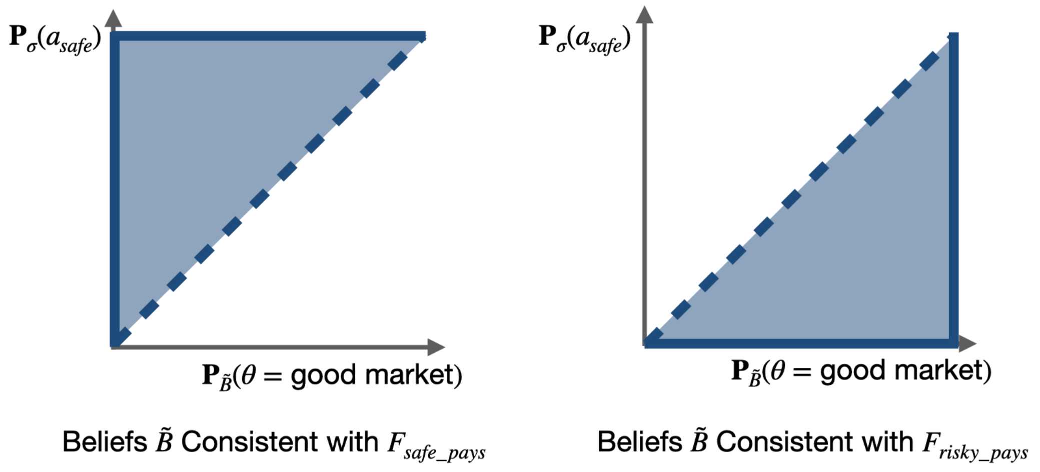

$\mathbf{P}_\sigma(a_{safe})$
$\mathbf{P}_{\tilde{B}}(\theta = \text{good market})$
Beliefs $\tilde{B}$ Consistent with F_{safe_pays}
$\mathbf{P}_\sigma(a_{safe})$
$\mathbf{P}_{\tilde{B}}(\theta = \text{good market})$
Beliefs $\tilde{B}$ Consistent with F_{risky_pays}

Figure 3.18: The beliefs consistent with F_{safe_pays} and F_{risky_pays}.

$\mathbb{P}_{\sigma,\tilde{w}}(\theta = \text{good market} \vee a = \text{risky}) < \mathbb{P}_{\sigma,\tilde{w}}(\theta = \text{bad market} \vee a = \text{safe})$, making F_{safe_pays} true.

While both F_{risky_pays} and F_{safe_pays} cannot hold for the same strategy in the world, either can be truthfully told to this agent, as the agent will go on to act in such a way as to choose a strategy which will causes the claim to be true. In this way, advice is often self-confirming.

3.4 When Strategy Referential and Agent Referential Claims Coincide

In the last section we showed that the strategy referential interpretation, though circularly defined, is often consistent, though with several nuances. In this section we will show that it also coincides with the agent referential claim. Taken together, since the agent-referential claim requires less nuance and is always well defined, the agent referential interpretation seems preferable in practice where the two will return the same results. Moreover, it shows that we do not need to resolve the ambiguity between the two for self referential claims to be useful in a wide range of situations.

Assuming that we are working with closed and unilaterally consistent self referential claims, the differences between the strategy referential and agent referential interpretation come down to three main factors:

- strategy referential claims can have weak preferences where agent-referential claims have strong preference
- strategy referential claims can choose sub-optimal equilibria when choosing between different consistent fixed points
- agent referential claims can prefer dominated strategies in when the self referential claim "assists suboptimal strategies"

The first of these cases was discussed in Section 3.3.1.2, since weakness is the cause of stubborn agents. In this section we will give examples of the other two: in Section 3.4.2 we will show how beliefs which assist suboptimal strategies can promote taking dominated strategies, and in Section 3.4.1 we will give an example where the strategy referential and agent referential interpretations differ due to the agent choosing a sub-optimal equilibrium. Finally, in section 3.4.3 we will show how, assuming the self-referential claim disallows beliefs which assist suboptimal strategies, the agent referential interpretation will find a strategy consistent with the strategy referential interpretation.

3.4.1 Equilibrium Selection

Consider a variation of "guess the bit", where it is more valuable to guess the bit if the bit happens to be a 0, returning 2 utility instead of one, which we call "biased guess the bit', as shown in figure 3.19. Then consider the agent with the prior $\tilde{B} = \{50\% : \{100\% : 0\}, 50\%\{100\% : 1\}\}$, and the self-referential claim $F_{BC} = \{(\sigma, \tilde{w})|\tilde{w} \in \text{argmax}_{\tilde{w}}\{\mathbb{E}_{a,\theta}[U(a, \theta]\}$, that is "you are likely to guess right". This claim and can be visualized in the usual way as shown in figure 3.19. Note that there are three consistent strategies which satisfy the strategy referential interpretation of this self referential claim, the strategy which deterministically chooses 0, and believes the world with the belief $\tilde{B} = \{100\% : \{100\% : 0\}\}$, and the strategy which deterministically chooses 1, and believes the world with the belief $\tilde{B} = \{100\% : \{100\% : 1\}\}$, and the strategy which chooses uniformly randomly $\sigma = \{50\% : 0, 50\% : 1\}$ and believes the bit is uniformly random $\tilde{B} = \{50\% : \{100\% : 0\}, 50\%\{100\% : 1\}\}$. Each of these are optimal strategies for their beliefs, and believe that the world was the best world for their current strategy, and can be seen as the intersection points of the visualization of F_BC with the best response function for guess the bit. More surprisingly, note that both of these are consistent with the prior $\tilde{B}$. That is, the self-referential belief update on F, is equally valid to return any one of these belief-strategy pairs, because they are all fixed-points of the strategy-referential Bayesian update.

The above situation satisfies the conditions of 5 and thus the agent referential interpretation will choose one of these fixed points as well. However while, the strategy referential interpretation would be satisfied with either equilibrium, the agent referential interpretation will always choose the best equilibrium, playing 0 deterministically. Thus for the two to be

	$\theta = 0$	$\theta = 1$
$a = 0$	2	0
$a = 1$	0	1

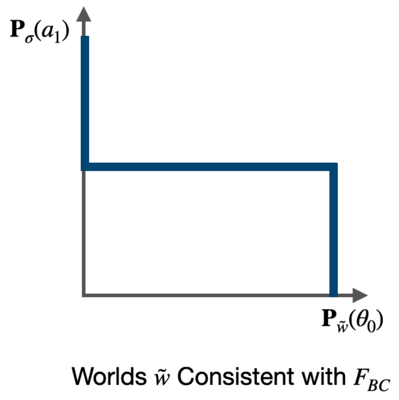


Figure 3.19: The payoff matrix for "biased guess the bit" and a visualization of F_{BC}

the same there would need to be an additional normative commitment on the part of the strategy referential agent to choose the better fixed point when there are two consistent fixed points

3.4.2 Agent Referential Beliefs Selecting Dominated Strategies

It is not always the case that the strategy referential and agent referential claims result in the same behavior even for continuous and unilaterally consistent self referential claims. A key case in which this occurs is when the agent referential interpretation justifies choosing a dominated strategy, which will never be chosen by the strategy referential interpretation.

For an example of this we can return to the twin prisoner's dilemma, described in Section 3.2.1, when introducing the two models. The reason why these two models produced different behavior was that the self referential claim $F_{same} = \{(\sigma, \tilde{w}) | \sigma = \tilde{w}\}$, assists suboptimal strategies. Specifically, note that the the action C in this environment is dominated, yet the agent referential interpretation chooses C. This is because the agent-referential update allows the agent to take them taking the action C as evidence that the other agent will also take the action C, thus adding a benefit to taking an action which, in a strategy referential sense, is strictly suboptimal.

More generally, we will say that a self-referential claim allows for *assisting suboptimal strategies* if and only if there is some strategies belief $\tilde{B}$ consistent with that claim and some σ, σ', such that σ' performs better than σ on $\tilde{B}(\sigma)$ and $\tilde{B}(\sigma')$, but σ gets more utility when each is measured on their own world. Or more formally:

$$U(\sigma, \tilde{B}(\sigma)) < U(\sigma', \tilde{B}(\sigma)) \tag{3.36}$$
$$U(\sigma, \tilde{B}(\sigma')) < U(\sigma', \tilde{B}(\sigma')) \tag{3.37}$$
$$U(\sigma, \tilde{B}) > U(\sigma', \tilde{B}) \tag{3.38}$$
$$\tag{3.39}$$

In this case, the F_{same} allows for assisting suboptimal strategies, since $\sigma^C = \{C : 100\%\}$ is dominated by $\sigma^D = \{D : 100\%\}$ under every belief, but σ^C is chosen over σ^D. An important point here is that this preference is due to an "assistance" that this policy gets in which it to tend to be in better worlds when it takes the dominated action, for reasons outside of that policies direct causal control. In the next section we will show that, when such cases are disallowed, the agent referential interpretation chooses a strategy compatible with the strategy referential interpretation.

3.4.3 Agent Referential Beliefs without Assisting Suboptimal Strategies

In this section we will show that, for any self referential claim which does not allow for assisting suboptimal strategies, the agent referential interpretation will choose a strategy consistent with the strategy referential interpretation. This means, in settings where the self referential belief is unilaterally consistent, closed, and does not allow for assisting suboptimal strategies, the agent referential interpretation is often safe to use even if one believes that the strategy referential interpretation is more proper, as they will arrive at the same result. This allows for avoiding the nuances of the strategy referential interpretation discussed in previous sections.

Theorem 5. *If the unilaterally consistent and closed self-referential claim F which does not allow for assisting suboptimal strategies, then the agent-referential interpretation of the claim will produce behavior which is consistent with the strategy-referential interpretation.*

Proof. Proceeding by contradiction, let $\sigma \in BR(\tilde{B})$, such that $\tilde{B}$ satisfies the agent referential interpretation of the self-referential claim F, but that there is no $\tilde{B}'(\sigma)$ such that $\sigma, \tilde{B}'(\sigma)$ which is consistent with the strategy referential interpretation of F, that is there is no $\tilde{B}'(\sigma)$ such that $\mathbb{P}_{\tilde{B}'(\sigma)}(F^S_\sigma) = 1$. Thus σ must not be a best response to the strategy referential interpretation of $\tilde{B}'$ at σ or formally $\sigma \notin BR(\tilde{B}(\sigma)^S)$, where $\tilde{B}(\sigma)^S$ refers to the belief in a fixed distribution of worlds $\tilde{B}(\sigma)$ independent of the counterfactual σ, as used in Theorem 4. We aim to show that this state of affairs implies that F assists suboptiomal strategies.

Note that there exists some σ' which is preferred to σ on all $\tilde{B}'(\sigma)$ which are consistent with the strategy referential interpretation of F at σ.

Consider a sequence of mixtures $\sigma^p = p\sigma + (1-p)\sigma'$ as p approaches 1 from below. For all of these mixtures σ^p must be more optimal than σ on all $\tilde{B}'(\sigma)$, since it is a mixture of σ

with something which is more optimal on $\tilde{B}'(\sigma)$. Moreover, consider the resulting sequence of $\tilde{B}(\sigma^p)^S$. The initial strategy σ must be more optimal than σ^p on all of each $\tilde{B}(\sigma^p)^S$ otherwise F would assist suboptimal strategies by definition.

Note that there is sequence of $\sigma^p, \tilde{B}(\sigma^p)$ which converges to $\sigma, \overline{\tilde{B}}(\sigma)^S$ for some $\overline{\tilde{B}}(\sigma)^S$, guaranteed to exist by the Bolzano–Weierstrass theorem. Since the sequence $\sigma^p, \tilde{B}(\sigma^p)$ converges to $\sigma, \overline{\tilde{B}}(\sigma)^S$, by the closedness of F we know that $\sigma, \overline{\tilde{B}}(\sigma)^S \in F$. Thus by the definition of σ', σ' must outperform σ on $\overline{\tilde{B}}(\sigma)^S$. We will show that this is not the case, thus finding a contradiction.

To show that σ' does not outperform σ on $\overline{\tilde{B}}(\sigma)^S$ we will show that for all p $\overline{\tilde{B}}(\sigma^p)^S$ is at least a constant distance away from any belief on which σ' outperforms σ. To show this, let $\tilde{B}''(\sigma)^S$ be some arbitrary belief on which σ^p is strictly preferred to σ. We will show that $\overline{\tilde{B}}(\sigma^p)^S$, for all p must be far from $\tilde{B}''(\sigma)^S$, and thus cannot converge to $\tilde{B}''(\sigma)^S$. Thus achieving our contradiction.

Consider the difference vector between the $\tilde{B}''(\sigma)^S$ and $\tilde{B}(\sigma^p)^S$, $\tilde{B}^{\Delta,S} = \tilde{B}''(\sigma)^S - \tilde{B}(\sigma^p)^S$. This allows us to re-express the difference in expected values of σ and σ^p on $\tilde{B}(\sigma^p)^S$ in terms of the performance of each on $\tilde{B}''(\sigma)^S$ and the difference vector. Since σ^p outperforms σ on $\tilde{B}''(\sigma)^S$ by p times some constant, it must be that σ must outperforms σ^p on the difference vector by at least p times that constant. We can then lower bound the performance difference between σ^p and σ by the highest possible difference between the performance of two actions, B, the probability p of the two actions being from different distributions, and the 1 norm of the difference vector. We then note that the p term divides out, so for σ^p to outperform σ on $\tilde{B}(\sigma^p)^S$ the difference between the two beliefs must be at least $\frac{C}{B}$. Since this is independent of p, this must persist in the limit, so $\overline{\tilde{B}}(\sigma^p)^S$ is at least a constant distance away from any belief on which σ^p is strictly preferred to σ, arriving at our contradiction. Or going through these same steps as equations:

$$
\begin{aligned}
U(\sigma^p, \tilde{B}(\sigma^p)^S) - U(\sigma, \tilde{B}(\sigma^p)^S) &= \mathop{\mathbb{E}}_{\substack{a \sim \sigma^p \\ \theta \sim \tilde{B}(\sigma^p)^S}} [U(a,\theta)] - \mathop{\mathbb{E}}_{\substack{a \sim \sigma \\ \theta \sim \tilde{B}(\sigma^p)^S}} [U(a,\theta)] \\
&= \mathop{\mathbb{E}}_{\substack{a \sim \sigma^p \\ a' \sim \sigma \\ \theta \sim \tilde{B}(\sigma^p)^S}} [U(a,\theta) - U(a',\theta)] \\
&= \mathop{\mathbb{E}}_{\substack{a \sim \sigma^p \\ a' \sim \sigma \\ \theta \sim \tilde{B}''(\sigma)^S}} [U(a,\theta) - U(a',\theta)] - \mathop{\mathbb{E}}_{\substack{a \sim \sigma^p \\ a' \sim \sigma \\ \theta \sim \tilde{B}^{\Delta,S}}} [U(a,\theta) - U(a',\theta)] \\
&= pC - \mathop{\mathbb{E}}_{\substack{a \sim \sigma^p \\ a' \sim \sigma \\ \theta \sim \tilde{B}^{\Delta,S}}} [U(a,\theta) - U(a',\theta)] \\
&= pC - \sum_{\theta \in \Theta} P_{\tilde{B}^{\Delta,S}}(\theta) \mathop{\mathbb{E}}_{\substack{a \sim \sigma^p \\ a' \sim \sigma}} [U(a,\theta) - U(a',\theta)] \\
&\geq pC - pB \sum_{\theta \in \Theta} P_{\tilde{B}^{\Delta,S}}(\theta) \\
&= pC - pB \sum_{\theta \in \Theta} (P_{\tilde{B}''(\sigma)^S}(\theta) - P_{\tilde{B}(\sigma^p)^S}(\theta)) \\
&\geq pC - pB||(\tilde{B}''(\sigma)^S - \tilde{B}(\sigma^p)^S||_{L1} \\
&= p(C - B||(\tilde{B}''(\sigma)^S - \tilde{B}(\sigma^p)^S||_{L1})
\end{aligned}
$$

.

Note that the in the above we are abusing the expectation and probability notation by using it over $\tilde{B}^{\Delta,S}$ which is not a distribution as it does not sum to 1. However, when defined in the natural way the proof works as shown.

Thus if $U(\sigma^p, \tilde{B}(\sigma^p)^S) - U(\sigma, \tilde{B}(\sigma^p)^S) \leq 0$, we have:

$$||(\tilde{B}''(\sigma)^S - \tilde{B}(\sigma^p)^S||_{L1} \geq \frac{C}{B}$$

Note that $\frac{C}{B}$ does not depend on p, thus as $\tilde{B}(\sigma^p)^S$ converges to $\overline{\overline{B}}(\sigma^p)^S$ it does not approach $\tilde{B}''(\sigma)^S$ as desired.

□

Thus, we have shown that not only is the strategy referential interpretation often consistent, but in fact the two interpretations are often the same.

3.5 Conclusions

We introduced two interpretations of self-referential claims, the strategy referential interpretation and agent referential interpretation, formalizing how we can extend the standard

Bayesian model to include the ability to update on self-referential claims. We discussed the differences between these two interpretations, showing that the two differ when there is a claim of assistive dominance, and that strategy referential claims can often be inconsistent or exhibit self-fulfilling prophecies which require additional normative commitments to avoid.

However, despite these differences, we showed that both interpretations of self-referential claims otherwise have the same effect. Once one disallows situations of assistive dominance, and add normative commitments to act as a fixed point and choose the best of self-fulfilling prophecies, the agent referential and strategy referential policies coincide. Thus we have shown that, for many applications, self-referential claims can be used without committing to either the strategy referential or agent referential interpretation.

Moreover, we have demonstrated that self-referential claims offer powerful new affordances to the AI designer. We have shown through the engine example that, by specifying the self-referential claim directly rather than requiring that it be derived from other specified beliefs, the designer can effectively specify claims that otherwise would be prohibitively difficult. Thus we have shown that allowing self-referential claims to be directly specified, rather than solely inferred can be a powerful affordance for the AI designer while maintaining consistency with the rest of the Bayesian framework.

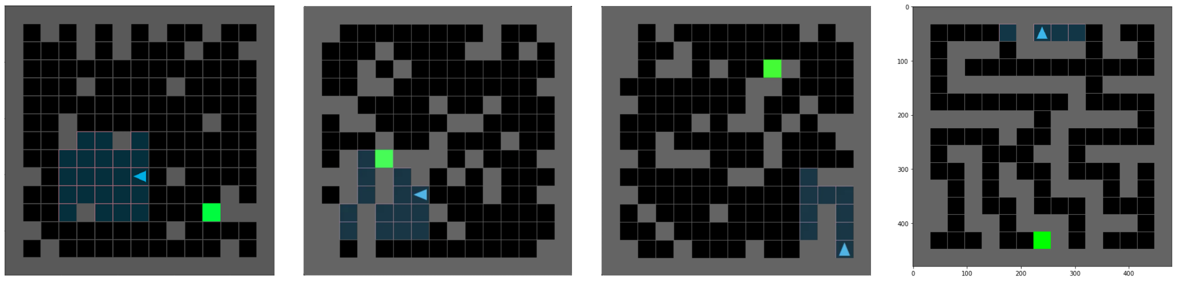
Chapter 4

PAIRED

Many reinforcement learning problems require designing a distribution of tasks and environments that can be used to evaluate and train effective policies. This is true for a diverse array of methods including transfer learning (e.g., [3, 86, 62, 79]), robust RL (e.g., [5, 30, 54]), unsupervised RL (e.g., [23]), and emergent complexity (e.g., [74, 84, 83]). For example, suppose we wish to train a robot in simulation to pick up objects from a bin in a real-world warehouse. There are many possible configurations of objects, including objects being stacked on top of each other. We may not know *a priori* the typical arrangement of the objects, but can naturally describe the simulated environment as having a distribution over the object positions.

However, designing an appropriate distribution of environments is challenging. The real world is complicated, and correctly enumerating all of the edge cases relevant to an application could be impractical or impossible. Even if the developer of the RL method knew every edge case, specifying this distribution could take a significant amount of time and effort. We want to automate this process.

In order for this automation to be useful, there must be a way of specifying the domain of environments in which the policy should be trained, without needing to fully specify the distribution. In our approach, developers only need to supply an *underspecified environment*: an environment which has free parameters which control its features and behavior. For instance, the developer could give a navigation environment in which the agent's objective is to navigate to the goal and the free parameters are the positions of obstacles. Our method will then construct distributions of environments by providing a distribution of settings of these free parameters; in this case, positions for those blocks. We call this problem of taking the underspecified environment and a policy, and producing an interesting distribution of fully specified environments in which that policy can be further trained *Unsupervised Environment Design* (UED). We formalize unsupervised environment design in Section 4.2, providing a characterization of the space of possible approaches which subsumes prior work. After training a policy in a distribution of environments generated by UED, we arrive at an updated policy and can use UED to generate more environments in which the updated policy can be trained. In this way, an approach for UED naturally gives rise to an approach



(a) Randomization (b) Minimax Adversary (c) PAIRED (ours) (d) Transfer task

Figure 4.1: An agent learns to navigate an environment where the position of the goal and obstacles is an underspecified parameter. If trained using domain randomization to randomly choose the obstacle locations (a), the agent will fail to generalize to a specific or complex configuration of obstacles, such as a maze (d). Minimax adversarial training encourages the adversary to create impossible environments, as shown in (b). In contrast, Protagonist Antagonist Induced Regret Environment Design (PAIRED), trains the adversary based on the difference between the reward of the agent (protagonist) and the reward of a second, antagonist agent. Because the two agents are learning together, the adversary is motivated to create a curriculum of difficult but achievable environments tailored to the agents' current level of performance (c). PAIRED facilitates learning complex behaviors and policies that perform well under zero-shot transfer to challenging new environments at test time.

for Unsupervised Curriculum Design (UCD). This method can be used to generate capable policies through increasingly complex environments targeted at the policy's current abilities.

Two prior approaches to UED are domain randomization, which generates fully specified environments uniformly randomly regardless of the current policy (e.g., [31, 62, 79]), and adversarially generating environments to minimize the reward of the current policy; *i.e.* minimax training (e.g., [54, 48, 81, 35]). While each of these approaches have their place, they can each fail to generate any interesting environments. In Figure 4.1 we show examples of maze navigation environments generated by each of these techniques. Uniformly random environments will often fail to generate interesting structures; in the maze example, it will be unlikely to generate walls (Figure 4.1a). On the other extreme, a minimax adversary is incentivized to make the environments completely unsolvable, generating mazes with unreachable goals (Figure 4.1b). In many environments, both of these methods fail to generate structured and solvable environments. We present a middle ground, generating environments which maximize regret, which produces difficult but solvable tasks (Figure 4.1c). Our results show that optimizing regret results in agents that are able to perform difficult transfer task (Figure 4.1d), which are not possible using the other two techniques.

We propose a novel adversarial training technique which naturally solves the problem of the adversary generating unsolvable environments by introducing an *antagonist* which is allied with the *environment-generating adversary*. For the sake of clarity, we refer to the primary agent we are trying to train as the *protagonist*. The environment adversary's goal is to design environments in which the antagonist achieves high reward and the protagonist receives low reward. If the adversary generates unsolvable environments, the antagonist and protagonist would perform the same and the adversary would get a score of zero, but if the adversary finds environments the antagonist solves and the protagonist does not solve, the adversary achieves a positive score. Thus, the environment adversary is incentivized to create challenging but *feasible* environments, in which the antagonist can outperform the protagonist. Moreover, as the protagonist learns to solves the simple environments, the antagonist must generate more complex environments to make the protagonist fail, increasing the complexity of the generated tasks and leading to automatic curriculum generation.

We show that when both teams reach a Nash equilibrium, the generated environments have the highest regret, in which the performance of the protagonist's policy most differs from that of the optimal policy. Thus, we call our approach *Protagonist Antagonist Induced Regret Environment Design (PAIRED)*. Our results demonstrate that compared to domain randomization, minimax adversarial training, and population-based minimax training (as in Wang et al. [84]), PAIRED agents learn more complex behaviors, and have higher zero-shot transfer performance in challenging, novel environments.

In summary, our main contributions are: a) formalizing the problem of Unsupervised Environment Design (UED) (Section 4.2), b) introducing the PAIRED Algorithm (Section 4.3), c) demonstrating PAIRED leads to more complex behavior and more effective zero-shot transfer to new environments than existing approaches to environment design (Section 4.4), and d) characterizing solutions to UED and connecting the framework to classical decision theory (Section 4.2).

4.1 Related Work

When proposing an approach to UED we are essentially making a decision about which environments to prioritize during training, based on an underspecified environment set provided by the designer. Making decisions in situations of extreme uncertainty has been studied in decision theory under the name "decisions under ignorance" [53] and is in fact deeply connected to our work, as we detail in Section 4.2. Milnor [46] characterize several of these approaches; using Milnor's terminology, we see domain randomization [31] as "the principle of insufficient reason" proposed by Laplace, minimax adversarial training [35] as the "maximin principle" proposed by Wald [82], and our approach, PAIRED, as "minimax regret" proposed by Savage [63]. Savage's approach was built into a theory of making sequential decisions which minimize worst case regret [61, 37], which eventually developed into modern literature on minimizing worst case regret in bandit settings [12].

Multi-agent training has been proposed as a way to automatically generate curricula in RL [38, 40, 22]. Competition can drive the emergence of complex skills, even some that the developer did not anticipate [6, 21]. Asymmetric Self Play (ASP) [74] trains a demonstrator agent to complete the simplest possible task that a learner cannot yet complete, ensuring that the task is, in principle, solvable. In contrast, PAIRED is significantly more general because it generates complex new environments, rather than trajectories within an existing, limited environment. Campero et al. [14] study curriculum learning in a similar environment to the one under investigation in this paper. They use a teacher to propose goal locations to which the agent must navigate, and the teacher's reward is computed based on whether the agent takes more or less steps than a threshold value. To create a curriculum, the threshold is linearly increased over the course of training. Bontrager and Togelius [8] procedurally generate environments based on the agent's value estimate. POET [84, 83] uses a population of minimax (rather than minimax *regret*) adversaries to generate the terrain for a 2D walker. However, POET [84] requires generating many new environments, testing all agents within each one, and discarding environments based on a manually chosen reward threshold, which wastes a significant amount of computation. In contrast, our minimax regret approach automatically learns to tune the difficulty of the environment by comparing the protagonist and antagonist's scores. In addition, these works do not investigate whether adversarial environment generation can provide enhanced generalization when agents are transferred to new environments, as we do in this paper.

Environment design has also been investigated within the symbolic AI community. The problem of modifying an environment to influence an agent's decisions is analyzed in Zhang, Chen, and Parkes [87], and was later extended through the concept of Value-based Policy Teaching [88]. Keren et al. [34, 33] use a heuristic best-first search to redesign an MDP to maximize expected value.

We evaluated PAIRED in terms of its ability to learn policies that transfer to unseen, hand-designed test environments. One approach to produce policies which can transfer between environments is unsupervised RL (e.g., [65, 67]). Gupta et al. [23] propose an unsupervised meta-RL technique that computes minimax regret against a diverse task distribution learned with the DIAYN algorithm [19]. However, this algorithm does not learn to modify the environment and does not adapt the task distribution based on the learning progress of the agent. PAIRED provides a curriculum of increasingly difficult environments and allows us to provide a theoretical characterization of when minimax regret should be preferred over other approaches.

The robust control literature has made use of both minimax and regret objectives. Minimax training has been used in the controls literature [73, 4, 89], the Robust MDP literature [5, 30, 50, 77], and more recently, through the use of adversarial RL training [54, 81, 48]. Unlike our algorithm, minimax adversaries have no incentive to guide the learning progress of the agent and can make environments arbitrarily hard or unsolvable. Ghavamzadeh, Petrik, and Chow [20] minimize the regret of a model-based policy against a safe baseline to ensure safe policy improvement. Regan and Boutilier [60, 59] study Markov Decision Processes (MDPs) in which the reward function is not fully specified, and use minimax regret as an objective to guide the elicitation of user preferences. While these approaches use similar theoretical techniques to ours, they use analytical solution methods that do not scale to the type of deep learning approach used in this paper, do not attempt to learn to generate environments, and do not consider automatic curriculum generation.

Domain randomization (DR) [31] is an alternative approach in which a designer specifies a set of parameters to which the policy should be robust [62, 79, 78]. These parameters are then randomly sampled for each episode, and a policy is trained that performs well on average across parameter values. However, this does not guarantee good performance on a specific, challenging configuration of parameters. While DR has had empirical success [2], it requires careful parameterization and does not automatically tailor generated environments to the current proficiency of the learning agent. Mehta et al. [44] propose to enhance DR by learning which parameter values lead to the biggest decrease in performance compared to a reference environment.

4.2 Unsupervised Environment Design

The goal of this work is to construct a policy that performs well across a large set of environments. We train policies by starting with an initially random policy, generating environments based on that policy to best suit its continued learning, train that policy in the generated environments, and repeat until the policy converges or we run out of resources. We then test the trained policies in a set of challenging transfer tasks not provided during training. In this section, we will focus on the environment generation step, which we call *unsupervised environment design* (UED). We will formally define UED as the problem of using an underspecified environment to produce a distribution over fully specified environments, which supports the continued learning of a particular policy. To do this, we must formally define fully specified and underspecified environments, and describe how to create a distribution of environments using UED. We will end this section by proposing minimax regret as a novel approach to UED.

We will model our fully specified environments with a Partially Observable Markov Decision Process (POMDP), which is a tuple $\langle A, O, S, \mathcal{T}, \mathcal{I}, \mathcal{R}, \gamma \rangle$ where A is a set of actions, O is a set of observations, S is a set of states, $\mathcal{T} : S \times A \rightarrow \mathbf{\Delta}(S)$ is a transition function, $\mathcal{I} : S \rightarrow O$ is an observation (or inspection) function, $\mathcal{R} : S \rightarrow \mathbb{R}$, and γ is a discount factor. We will define the utility as $U(\pi) = \sum_{i=0}^{T} r_t \gamma^t$, where T is a horizon.

To model an underspecified environment, we propose the Underspecified Partially Observable Markov Decision Process (UPOMDP) as a tuple $\mathcal{M} = \langle A, O, \Theta, S^{\mathcal{M}}, \mathcal{T}^{\mathcal{M}}, \mathcal{I}^{\mathcal{M}}, \mathcal{R}^{\mathcal{M}}, \gamma \rangle$. The only difference between a POMDP and a UPOMDP is that a UPOMDP has a set Θ representing the free parameters of the environment, which can be chosen to be distinct at each time step and are incorporated into the transition function as $\mathcal{T}^{\mathcal{M}} : S \times A \times \Theta \rightarrow \mathbf{\Delta}(S)$. Thus a possible setting of the environment is given by some trajectory of environment parameters $\vec{\theta}$. As an example UPOMDP, consider a simulation of a robot in which $\vec{\theta}$ are additional forces which can be applied at each time step. A setting of the environment $\vec{\theta}$ can be naturally combined with the underspecified environment $\mathcal{M}$ to give a specific POMDP, which we will denote $\mathcal{M}_{\vec{\theta}}$.

In UED, we want to generate a distribution of environments in which to continue training a policy. We would like to curate this distribution of environments to best support the continued learning of a particular agent policy. As such, we can propose a solution to UED

Table 4.1: Example Environment Policies

UED Technique	Environment Policy	Decision Rule
Domain Randomization [31, 62, 79]	$\Lambda^{DR}(\pi) = \mathcal{U}(\Theta^T)$	Insufficient Reason
Minimax Adversary [54, 81, 48]	$\Lambda^{M}(\pi) = \underset{\vec{\theta} \in \Theta^T}{\text{argmin}}\{U^{\vec{\theta}}(\pi)\}$	Maximin
PAIRED (ours)	$\Lambda^{MR}(\pi) = \{\overline{\theta}_\pi : \frac{c_\pi}{v_\pi}, \tilde{D}_\pi : \text{otherwise}\}$	Minimax Regret

Table 4.2: The environment policies corresponding to the three techniques for UED which we study, along with their corresponding decision rule. Where $\mathcal{U}(X)$ is a uniform distribution over X, $\tilde{D}_\pi$ is a baseline distribution, $\overline{\theta}_\pi$ is the trajectory which maximizes regret of π, v_π is the value above the baseline distribution that π achieves on that trajectory, and c_π is the negative of the worst-case regret of π, normalized so that $\frac{c_\pi}{v_\pi}$ is between 0 and 1. This is described in full in the appendix.

by specifying some *environment policy*, $\Lambda : \Pi \rightarrow \boldsymbol{\Delta}(\Theta^T)$ where Π is the set of possible policies and Θ^T is the set of possible sequences of environment parameters. In Table 4.2, we see a few examples of possible choices for environment policies, as well as how they correspond to previous literature. Each of these choices also have a corresponding decision rule used for making *decisions under ignorance* [53]. Each decision rule can be understood as a method for choosing a policy given an underspecified environment. This connection between UED and decisions under ignorance extends further than these few decision rules. In the Appendix we will make this connection concrete and show that, under reasonable assumptions, UED and decisions under ignorance are solving the same problem.

PAIRED can be seen as an approach for approximating the environment policy, Λ^{MR}, which corresponds to the decision rule minimax regret. One useful property of minimax regret, which is not true of domain randomization or minimax adversarial training, is that whenever a task has a sufficiently well-defined notion of success and failure it chooses policies which succeed. By minimizing the worst case difference between what is achievable and what it achieves, whenever there is a policy which ensures success, it will not fail where others could succeed.

Theorem 6. *Suppose that all achievable rewards fall into one of two class of outcomes labeled* ***SUCCESS*** *giving rewards in* $[\boldsymbol{S}_{min}, \boldsymbol{S}_{max}]$ *and* ***FAILURE*** *giving rewards in* $[\boldsymbol{F}_{min}, \boldsymbol{F}_{max}]$*, such that* $\boldsymbol{F}_{min} \leq \boldsymbol{F}_{max} < \boldsymbol{S}_{min} \leq \boldsymbol{S}_{max}$*. In addition assume that the range of possible rewards in either class is smaller than the difference between the classes so we have* $\boldsymbol{S}_{max} - \boldsymbol{S}_{min} < \boldsymbol{S}_{min} - \boldsymbol{F}_{max}$ *and* $\boldsymbol{F}_{max} - \boldsymbol{F}_{min} < \boldsymbol{S}_{min} - \boldsymbol{F}_{max}$*. Further suppose that there is a policy* π *which succeeds on any* $\vec{\theta}$ *whenever success is possible. Then minimax regret will choose a policy which has that property.*

The proof of this property, and examples showing how minimax and domain randomization fail to have this property, are in the Appendix. In the next section we will formally introduce PAIRED as a method for approximating the minimax regret environment policy, Λ^{MR}.

4.3 Protagonist Antagonist Induced Regret Environment Design (PAIRED)

Here, we describe how to approximate minimax regret, and introduce our proposed algorithm, Protagonist Antagonist Induced Regret Environment Design (PAIRED). Regret is defined as the difference between the payoff obtained for a decision, and the optimal payoff that could have been obtained in the same setting with a different decision. In order to approximate regret, we use the difference between the payoffs of two agents acting under the same environment conditions. Assume we are given a fixed environment with parameters $\vec{\theta}$, a fixed policy for the protagonist agent, π^P, and we then train a second antagonist agent, π^A, to optimality in this environment. Then, the difference between the reward obtained by the antagonist, $U^{\vec{\theta}}(\pi^A)$, and the protagonist, $U^{\vec{\theta}}(\pi^P)$, is the regret:

$$\textsc{Regret}^{\vec{\theta}}\left(\pi^P, \pi^A\right) = U^{\vec{\theta}}\left(\pi^A\right) - U^{\vec{\theta}}\left(\pi^P\right) \tag{4.1}$$

In PAIRED, we introduce an environment adversary that learns to control the parameters of the environment, $\vec{\theta}$, to maximize the regret of the protagonist against the antagonist. For each training batch, the adversary generates the parameters of the environment, $\vec{\theta} \sim \tilde{\Lambda}$, which both agents will play. The adversary and antagonist are trained to maximize the regret as computed in Eq. 4.1, while the protagonist learns to minimize regret. This procedure is shown in Algorithm 1. The code for PAIRED and our experiments is available in open source at `https://github.com/google-research/google-research/tree/master/social_rl/`. We note that our approach is agnostic to the choice of RL technique used to optimize regret.

To improve the learning signal for the adversary, once the adversary creates an environment, both the protagonist and antagonist generate several trajectories within that same environment. This allows for a more accurate approximation the minimax regret as the difference between the maximum reward of the antagonist and the average reward of the protagonist over all trajectories: $\textsc{Regret} \approx \max_{\tau^A} U^{\vec{\theta}}(\tau^A) - \mathbb{E}_{\tau^P}[U^{\vec{\theta}}(\tau^P)]$. We have found this reduces noise in the reward and more accurately rewards the adversary for building difficult but solvable environments.

Algorithm 1: PAIRED.

```
Randomly initialize Protagonist $\pi^P$, Antagonist $\pi^A$, and Adversary $\tilde{\Lambda}$;
while not converged do
    Use adversary to generate environment parameters: $\vec{\theta} \sim \tilde{\Lambda}$. Use to create
     POMDP $\mathcal{M}_{\vec{\theta}}$.
    Collect Protagonist trajectory $\tau^P$ in $\mathcal{M}_{\vec{\theta}}$. Compute: $U^{\vec{\theta}}(\pi^P) = \sum_{i=0}^{T} r_t \gamma^t$
    Collect Antagonist trajectory $\tau^A$ in $\mathcal{M}_{\vec{\theta}}$. Compute: $U^{\vec{\theta}}(\pi^A) = \sum_{i=0}^{T} r_t \gamma^t$
    Compute: $\textsc{Regret}^{\vec{\theta}}(\pi^P, \pi^A) = U^{\vec{\theta}}(\pi^A) - U^{\vec{\theta}}(\pi^P)$
    Train Protagonist policy $\pi^P$ with RL update and reward $R(\tau^P) = -\textsc{Regret}$
    Train Antagonist policy $\pi^A$ with RL update and reward $R(\tau^A) = \textsc{Regret}$
    Train Adversary policy $\tilde{\Lambda}$ with RL update and reward $R(\tau^{\tilde{\Lambda}}) = \textsc{Regret}$
end
```

At each training step, the environment adversary can be seen as solving a UED problem for the current protagonist, generating a curated set of environments in which it can learn. While the adversary is motivated to generate tasks beyond the protagonist's abilities, the regret can actually incentivize creating the easiest task on which the protagonist fails but the antagonist succeeds. This is because if the reward function contains any bonus for solving the task more efficiently (e.g. in fewer timesteps), the antagonist will get more reward if the task is easier. Thus, the adversary gets the most regret for proposing the easiest task which is outside the protagonist's skill range. Thus, regret incentivizes the adversary to propose tasks within the agent's "zone of proximal development" [15]. As the protagonist learns to solve the simple tasks, the adversary is forced to find harder tasks to achieve positive reward, increasing the complexity of the generated tasks and leading to automatic curriculum generation.

Though multi-agent learning may not always converge [42], we can show that, if each team in this game finds an optimal solution, the protagonist would be playing a minimax regret policy.

Theorem 7. *Let $(\pi^P, \pi^A, \vec{\theta})$ be in Nash equilibrium and the pair $(\pi^A, \vec{\theta})$ be jointly a best response to π^P. Then $\pi^P \in \underset{\pi^P \in \Pi^P}{\operatorname{argmin}}\{\underset{\pi^A, \vec{\theta} \in \Pi^A \times \Theta^T}{\operatorname{argmax}} \{\textsc{Regret}^{\vec{\theta}}(\pi^P, \pi^A)\}\}$.*

Proof. Let $(\pi^P, \pi^A, \vec{\theta})$ be in Nash equilibria and the pair $(\pi^A, \vec{\theta})$ is jointly a best response to π^P. Then we can consider $(\pi^A, \vec{\theta})$ as one player with policy which we will write as $\pi^{A+\vec{\theta}} \in \Pi \times \Theta^T$. Then $\pi^{A+\vec{\theta}}$ is in a zero sum game with π^P, and the condition that the pair $(\pi^A, \vec{\theta})$ is jointly a best response to π^P is equivalent to saying that $\pi^{A+\vec{\theta}}$ is a best response to π^P. Thus π^P and $\pi^{A+\vec{\theta}}$ form a Nash equilibria of a zero sum game, and the minimax theorem applies. Since the reward in this game is defined by Regret we have:

$$\pi^P \in \underset{\pi^P \in \Pi^P}{\operatorname{argmin}}\{\underset{\pi^{A+\vec{\theta}} \in \Pi \times \Theta^T}{\operatorname{argmax}} \{\textsc{Regret}^{\vec{\theta}}(\pi^P, \pi^A)\}\}$$

By the definition of $\pi^{A+\vec{\theta}}$ this proves the protagonist learns the minimax regret policy. $\square$

The proof gives us reason to believe that the iterative PAIRED training process in Algorithm 1 can produce minimax regret policies. In Appendix D we show that if the agents are in Nash equilibrium, without the coordination assumption, the protagonist will perform at least as well as the antagonist in every parameterization, and Appendix E.0.1 provides empirical results for alternative methods for approximating regret which break the coordination assumption. In the following sections, we will show that empirically, policies trained with Algorithm 1 exhibit good performance and transfer, both in generating emergent complexity and in training robust policies.

4.4 Experiments

The experiments in this section focus on assessing whether training with PAIRED can increase the complexity of agents' learned behavior, and whether PAIRED agents can achieve better or more robust performance when transferred to novel environments. To study these questions, we first focus on the navigation tasks shown in Figure 4.1. Section 4.4.2 presents results in continuous domains.

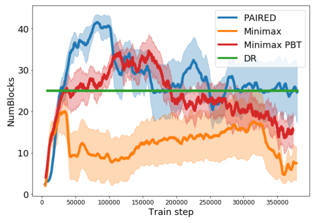
4.4.1 Partially Observable Navigation Tasks

Here we investigate navigation tasks (based on [16]), in which an agent must explore to find a goal (green square in Figure 4.1) while navigating around obstacles. The environments are partially observable; the agent's field of view is shown as a blue shaded area in the figure. To deal with the partial observability, we parameterize the protagonist and antagonist's policies using recurrent neural networks (RNNs). All agents are trained with PPO [66]. Further details about network architecture and hyperparameters are given in Appendix F.

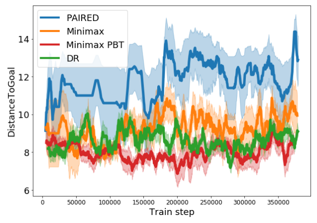
We train adversaries that learn to build these environments by choosing the location of the obstacles, the goal, and the starting location of the agent. The adversary's observations consist of a fully observed view of the environment state, the current timestep t, and a random vector $z \sim \mathcal{N}(0, I), z \in \mathbb{R}^D$ sampled for each episode. At each timestep, the adversary outputs the location where the next object will be placed; at timestep 0 it places the agent, 1 the goal, and every step afterwards an obstacle. Videos of environments being constructed and transfer performance are available at `https://www.youtube.com/channel/UCI6dkF8eNrCz6XiBJlV9fmw/videos`.

4.4.1.1 Comparison to Prior Methods

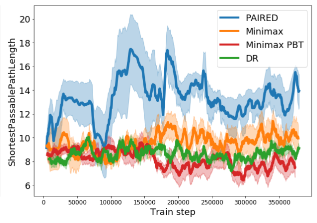
To compare to prior work that uses pure minimax training [54, 81, 48] (rather than minimax *regret*), we use the same parameterization of the environment adversary and protagonist, but simply remove the antagonist agent. The adversary's reward is $R(\Lambda) = -\mathbb{E}_{\tau^P}[U(\tau^P)]$. While a direct comparison to POET [84] is challenging since many elements of the POET



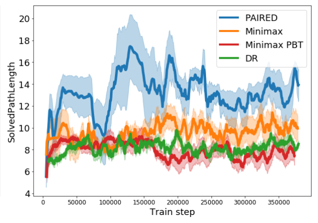
(a) Number of blocks



(b) Distance to goal



(c) Passable path length



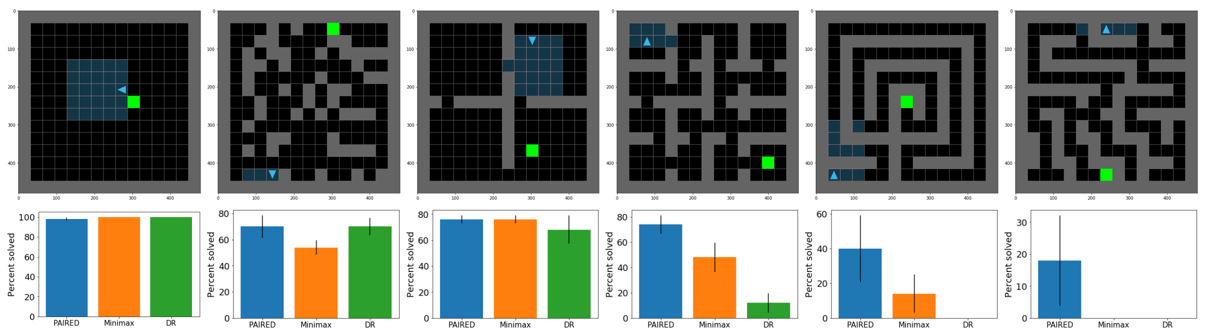
(d) Solved path length

Figure 4.2: Statistics of generated environments in terms of the number of blocks (a), distance from the start position to the goal (b), and the shortest path length between the start and the goal, which is zero if there is no possible path (c). The final plot shows agent learning in terms of the shortest path length of a maze successfully solved by the agents. Each plot is measured over five random seeds; error bars are a 95% CI. Domain randomization (DR) cannot tailor environment design to the agent's progress, so metrics remain fixed or vary randomly. Minimax training (even with populations of adversaries and agents) has no incentive to improve agent performance, so the length of mazes that agents are able to solve remains similar to DR (d). In contrast, PAIRED is the only method that continues to increase the passable path length to create more challenging mazes (c), producing agents that solve more complex mazes than the other techniques (d).

algorithm are specific to a 2D walker, the main algorithmic distinction between POET and minimax environment design is maintaining a population of environment adversaries and agents that are periodically swapped with each other. Therefore, we also employ a Population Based Training (PBT) minimax technique in which the agent and adversary are sampled from respective populations for each episode. This baseline is our closest approximation of POET [84]. To apply domain randomization, we simply sample the (x, y) positions of the agent, goal, and blocks uniformly at random. We sweep shared hyperparameters for all methods equally. Parameters for the emergent complexity task are selected to maximize the solved path length, and parameters for the transfer task are selected using a set of validation environments. Details are given in the appendix.

4.4.1.2 Emergent Complexity

Prior work [84, 83] focused on demonstrating emergent complexity as the primary goal, arguing that automatically learning complex behaviors is key to improving the sophistication of AI agents. Here, we track the complexity of the generated environments and learned behaviors throughout training. Figure 4.2 shows the number of blocks (a), distance to the goal (b), and the length of the shortest path to the goal (c) in the generated environments. The solved path length (d) tracks the shortest path length of a maze that the agent has completed successfully, and can be considered a measure of the complexity of the agent's learned behavior.



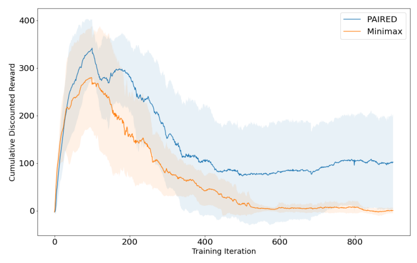
(a) Empty (b) 50 Blocks (c) 4 Rooms (d) 16 Rooms (e) Labyrinth (f) Maze

Figure 4.3: Percent successful trials in environments used to test zero-shot transfer, out of 10 trials each for 5 random seeds. The first two (a, b) simply test out-of-distribution generalization to a setting of the number of blocks parameter. Four Rooms (c) tests within-distribution generalization to a specific configuration that is unlikely to be generated through random sampling. The 16 Rooms (d), Labyrinth environment (e) and Maze (f) environments were designed by a human to be challenging navigation tasks. The bar charts show the zero-shot transfer performance of models trained with domain randomization (DR), minimax, or PAIRED in each of the environments. Error bars show a 95% confidence interval. As task difficulty increases, only PAIRED retains its ability to generalize to the transfer tasks.

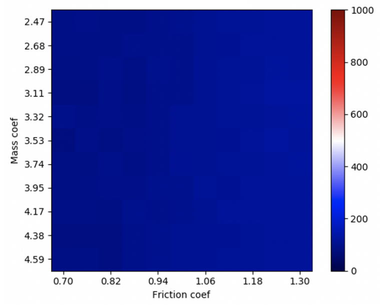
Domain randomization (DR) simply maintains environment parameters within a fixed range, and cannot continue to propose increasingly difficult tasks. Techniques based on minimax training are purely motivated to decrease the protagonist's score, and as such do not enable the agent to continue learning; both minimax and PBT obtain similar performance to DR. In contrast, PAIRED creates a curriculum of environments that begin with shorter paths and fewer blocks, but gradually increase in complexity based on both agents' current level of performance. As shown in Figure 4.2 (d), this allows PAIRED protagonists to learn to solve more complex environments than the other two techniques.

4.4.1.3 Zero-Shot Transfer

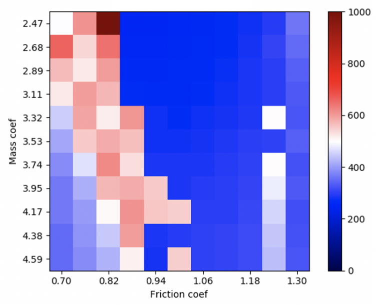
In order for RL algorithms to be useful in the real world, they will need to generalize to novel environment conditions that their developer was not able to foresee. Here, we test the zero-shot transfer performance on a series of novel navigation tasks shown in Figure 4.3. The first two transfer tasks simply use a parameter setting for the number of blocks that is out of the distribution (OOD) of the training environments experienced by the agents (analogous to the evaluation of [54]). We expect these transfer scenarios to be easy for all methods. We also include a more structured but still very simple Four Rooms environment, where the number of blocks is within distribution, but in an unlikely (though simple) configuration. As seen in the figure, this can usually be completed with nearly straight-line paths to the



(a) Reward



(b) Minimax Adversarial



(c) PAIRED

Figure 4.4: Training curves for PAIRED and minimax adversaries on the MuJoCo hopper, where adversary strength is scaled up over the first 300 iterations (a). While both adversaries reduce the agent's reward by making the task more difficult, the minimax adversary is incentivized to drive the agent's reward as low as possible. In contrast, the PAIRED adversary must maintain feasible parameters. When varying force and mass are applied at test time, the minimax policy performs poorly in cumulative reward (b), while PAIRED is more robust (c).

goal. To evaluate difficult transfer settings, we include the 16 rooms, Labyrinth, and Maze environments, which require traversing a much longer and more challenging path to the goal. This presents a much more challenging transfer scenario, since the agent must learn meaningful navigation skills to succeed in these tasks.

Figure 4.3 shows zero-shot transfer performance. As expected, all methods transfer well to the first two environments, although the minimax adversary is significantly worse in 50 Blocks. Varying the number of blocks may be a relatively easy transfer task, since partial observability has been shown to improve generalization performance in grid-worlds [85]. As the environments increase in difficulty, so does the performance gap between PAIRED and the prior methods. In 16 Rooms, Labyrinth, and Maze, PAIRED shows a large advantage over the other two methods. In the Labyrinth (Maze) environment, PAIRED agents are able to solve the task in 40% (18%) of trials. In contrast, minimax agents solve 10% (0.0%) of trials, and DR agents solve 0.0%. The performance of PAIRED on these tasks can be explained by the complexity of the generated environments; as shown in Figure 4.1c, the field of view of the agent (shaded blue) looks similar to what it might encounter in a maze, although the idea of a maze was never explicitly specified to the agent, and this configuration blocks is very unlikely to be generated randomly. These results suggest that PAIRED training may be better able to prepare agents for challenging, unknown test settings.

4.4.2 Continuous Control Tasks

To compare more closely with prior work on minimax adversarial RL [54, 81], we construct an additional experiment in a modified version of the MuJoCo hopper domain [80]. Here, the adversary outputs additional torques to be applied to each joint at each time step, $\vec{\theta}$. We benchmark PAIRED against unconstrained minimax training. To make minimax training feasible, the torque the adversary can apply is limited to be a proportion of the agent's strength, and is scaled from 0.1 to 1.0 over the course of 300 iterations. After this point, we continue training with no additional mechanisms for constraining the adversary. We pre-train the agents for 100 iterations, then test transfer performance to a set of mass and friction coefficients. All agents are trained with PPO [66] and feed-forward policies.

Figure 4.4a shows the rewards throughout training. We observe that after the constraints on the adversaries are removed after 300 iterations, the full-strength minimax adversary has the ability to make the task unsolvable, so the agent's reward is driven to zero for the rest of training. In contrast, PAIRED is able to automatically adjust the difficulty of the adversary to ensure the task remains solvable. As expected, Figure 4.4 shows that when constraints on the adversary are not carefully tuned, policies trained with minimax fail to learn and generalize to unseen environment parameters. In contrast, PAIRED agents are more robust to unseen environment parameters, even without careful tuning of the forces that the adversary is able to apply.

4.5 Conclusions

We develop the framework of Unsupervised Environment Design (UED), and show its relevance to a range of RL tasks, from learning increasingly complex behavior or more robust policies, to improving generalization to novel environments. In environments like games, for which the designer has an accurate model of the test environment, or can easily enumerate all of the important cases, using UED may be unnecessary. However, UED could provide an approach to building AI systems in many of the ambitious uses of AI in real-world settings which are difficult to accurately model.

We have shown that tools from the decision theory literature can be used to characterize existing approaches to environment design, and motivate a novel approach based on minimax regret. Our algorithm, which we call Protagonist Antagonist Induced Regret Environment Design (PAIRED), avoids common failure modes of existing UED approaches, and is able to generate a curriculum of increasingly complex environments. Our results demonstrate that PAIRED agents learn more complex behaviors, and achieve higher zero-shot transfer performance in challenging, novel environments.

Chapter 5

Conclusion

In this thesis, we have shown how treating problem specifications as acts of communication can lead to a deeper understanding of how to efficiently and effectively specify problems, as well as fundamental insights into how we would want our AI systems to make decisions. We begin from a perspective reminiscent of rational speech acts, understanding the utility of a problem specification as the utility that the act of specifying and solving that specification would have for the AI designer. Our work has demonstrated that this simple premise leads to surprising insights.

Our work lays the foundations for a theoretical framework for problem specification, the specification design problem. Moreover, we showed that analyzing this problem can be fruitful. The theory allows us to provide concrete recommendations for AI designers when designing their problem specifications through the justification value of information which describes what should be specified, or when it is better to leave something unspecified to reduce specification costs or the risk of specification error.

Moreover, we propose an extension to the standard Bayesian framework of AI, through the ability to directly specify and update on self-referential claims. We demonstrate in Chapter 3 how directly specifying a self-referential claim can be significantly more efficient than relying on only direct empirical facts. Though this increases the need for decision-theoretic nuance, broadening the scope of disagreements between CDT and EDT by having to distinguish between the strategy referential and the agent referential interpretations of self-referential claims, we were able to show that this nuance can be largely avoided in a broad range of settings. Thus, inspired by the specification design problem framework, we were able to construct a rigorous model for self-referential claims laying the foundations for AI systems that can handle such claims in the future.

Finally, in Chapter 4, we introduce Unsupervised Environment Design (UED), and the algorithm PAIRED, which designs environments which result in minimax regret policies in equilibrium. We showed that the framework of unsupervised environment design is expressive enough to represent a broad class of decision theories, in effect equivalent to the set of policies consistent with a strategy referential interpretation of a self-referential claim. Moreover, the PAIRED algorithm, since it produces minimax regret policies in equilibrium, gives the implementation necessary to apply the justification value of assumptions discussed in Chapter 2. Overall, Unsupervised Environment Design, and PAIRED allow the application of approaches to specification design problems at the scale of modern RL systems.

5.1 Future Work

The framework we have made for justifying, constructing, and optimizing problem specifications leads to many exciting directions for future work through building on the formalization of specification design problems, self-referential claims, and unsupervised environment design.

Building on Specification Design Problems Further analysis of the specification design problem offers a promising direction toward the resolution of many of the overly vague, and often normative problems facing AI such as optimal strategies for exploration, equilibrium selection, fault tolerance, and being robust to unknown unknowns. While such problems are difficult to tackle, because of their underspecified nature, the specification design problem allows the analysis of this underspecification explicitly, and thus allows these problems to be approached completely formally. Thus there is an exciting opportunity to study the implicit assumptions underlying current approaches to these problems and to propose new solutions which work under broader assumptions.

Moreover, there are many promising directions for building directly on the justification value of assumptions. Not only could one use this framework to study current approaches to specification and the limitations of such approaches, one could use this approach to define better problem specifications that specify only the critical pieces of information. More ambitiously, we could aim to construct an efficient and scalable approach to implement the minimax regret language as the universal approach to justifiable problem specifications, as PAIRED aims to do. Reducing the in-practice problem of problem specification to determining the best assumptions to use for a PAIRED-like algorithm.

The justification value of assumptions itself could be extended to include interactive protocols, for situations where the designer is unsure of the value of different assumptions. Thus the human AI designer must ask the AI agent, who understands more about how to solve the domain, for information about the practical considerations of the domain, before the AI designer would know what is most effective to specify. Such protocols could be built off of similar protocols in communication theory and would serve to make specifications more robust and efficient.

Building on Self Referential Claims Simultaneous, there are several useful directions towards building AI systems that properly handle self-referential claims, most directly by developing variants of common AI approaches such as direct planning, policy-gradient, or Q-learning algorithms that perform optimally with respect to self-referential claims. Such algorithms would allow us to make systems that can naturally incorporate advice and can be made more directly aware of their failures or limitations.

Moreover, self-referential claims not only give a new perspective to the existing debate between CDT v.s. EDT decision theory, but also a perspective on other decision-theoretic problems like decisions under ignorance, risk aversion, unknown unknowns, and wicked problems. From such a perspective, it may be possible to make dramatic progress in a number of decision-theoretic disagreements which have otherwise resisted solution.

Building on Unsupervised Environment Design Finally, Unsupervised Environment Design is a quickly growing field with countless interesting directions. As an approach for solving novel problem specifications, it is quite versatile, as solving a new problem specification, as long as it is not assistive to dominance, is as simple as pointing the environment designer at a new design objective. The most prevalent variant of UED algorithms, using regret-based environment designers, is perfect for implementing the universal specification language and applying the theory of justifiable problem specifications, thus integrating insights from these theories has promised to give the AI designer better control over the trained policies.

As such, any progress in making UED algorithms more efficient and scalable is progress that can be used to solve robust and efficient specifications. Thus, following the recent trends of progress in UED being driven by better optimization algorithms for the environment optimizer [32, 51], it seems highly impactful to be able to utilize the high-dimensional optimization algorithms behind work such as GPT-3 and Dall-E which has been effective in generating complex high-dimensional images and text which exhibit the structural complexity of the real world[11, 55]. If these tools could be harnessed to generate complex and realistic training environments, efficiently enough for those environments to be used to train RL systems, then UED could become a pivotal piece of every RL pipeline. As such, UED not only gives the designer more control over the resulting policy, and allows us to adapt our algorithms towards better specifications as we discover them, but is also a critical piece to making capable and effective RL policies at scale.

Bibliography

[1] Dario Amodei et al. “Concrete problems in AI safety”. In: *arXiv preprint arXiv:1606.06565* (2016).

[2] OpenAI: Marcin Andrychowicz et al. “Learning dexterous in-hand manipulation”. In: *The International Journal of Robotics Research* 39.1 (2020), pp. 3–20.

[3] Rika Antonova et al. “Reinforcement learning for pivoting task”. In: *arXiv preprint arXiv:1703.00472* (2017).

[4] Karl Johan Åström. “Theory and applications of adaptive control—a survey”. In: *Automatica* 19.5 (1983), pp. 471–486.

[5] J Andrew Bagnell, Andrew Y Ng, and Jeff G Schneider. “Solving uncertain Markov decision processes”. In: (2001).

[6] Bowen Baker et al. “Emergent tool use from multi-agent autocurricula”. In: *arXiv preprint arXiv:1909.07528* (2019).

[7] Richard Bellman. “A Markovian decision process”. In: *Journal of mathematics and mechanics* (1957), pp. 679–684.

[8] Philip Bontrager and Julian Togelius. “Fully differentiable procedural content generation through generative playing networks”. In: *arXiv preprint arXiv:2002.05259* (2020).

[9] George EP Box. “Science and statistics”. In: *Journal of the American Statistical Association* 71.356 (1976), pp. 791–799.

[10] Noam Brown, Tuomas Sandholm, and Strategic Machine. “Libratus: The Superhuman AI for No-Limit Poker.” In: *IJCAI*. 2017, pp. 5226–5228.

[11] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[12] Sébastien Bubeck and Nicolo Cesa-Bianchi. “Regret analysis of stochastic and nonstochastic multi-armed bandit problems”. In: *arXiv preprint arXiv:1204.5721* (2012).

[13] Yuri Burda et al. “Exploration by random network distillation”. In: *arXiv preprint arXiv:1810.12894* (2018).

[14] Andres Campero et al. “Learning with AMIGo: Adversarially Motivated Intrinsic Goals”. In: *arXiv preprint arXiv:2006.12122* (2020).

[15] Seth Chaiklin et al. "The zone of proximal development in Vygotsky's analysis of learning and instruction". In: *Vygotsky's educational theory in cultural context* 1.2 (2003), pp. 39–64.

[16] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. *Minimalistic Gridworld Environment for OpenAI Gym.* `https://github.com/maximecb/gym-minigrid`. 2018.

[17] Michael Dennis et al. "Emergent complexity and zero-shot transfer via unsupervised environment design". In: *Advances in neural information processing systems* 33 (2020), pp. 13049–13061.

[18] Lauro Langosco Di Langosco et al. "Goal Misgeneralization in Deep Reinforcement Learning". In: *International Conference on Machine Learning.* PMLR. 2022, pp. 12004–12019.

[19] Benjamin Eysenbach et al. "Diversity is all you need: Learning skills without a reward function". In: *arXiv preprint arXiv:1802.06070* (2018).

[20] Mohammad Ghavamzadeh, Marek Petrik, and Yinlam Chow. "Safe policy improvement by minimizing robust baseline regret". In: *Advances in Neural Information Processing Systems.* 2016, pp. 2298–2306.

[21] Adam Gleave et al. "Adversarial Policies: Attacking Deep Reinforcement Learning". In: *International Conference on Learning Representations.* 2020.

[22] Alex Graves et al. "Automated curriculum learning for neural networks". In: *arXiv preprint arXiv:1704.03003* (2017).

[23] Abhishek Gupta et al. "Unsupervised meta-learning for reinforcement learning". In: *arXiv preprint arXiv:1806.04640* (2018).

[24] Dylan Hadfield-Menell et al. "Cooperative inverse reinforcement learning". In: *Advances in neural information processing systems* 29 (2016).

[25] Dylan Hadfield-Menell et al. "Inverse reward design". In: *Advances in neural information processing systems* 30 (2017).

[26] Dylan Hadfield-Menell et al. "The off-switch game". In: *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence.* 2017.

[27] Danny Hernandez and Tom B Brown. "Measuring the algorithmic efficiency of neural networks". In: *arXiv preprint arXiv:2005.04305* (2020).

[28] Christopher Hitchcock. "Conditioning, intervening, and decision". In: *Synthese* 193.4 (2016), pp. 1157–1176.

[29] Ronald A Howard. "Information value theory". In: *IEEE Transactions on systems science and cybernetics* 2.1 (1966), pp. 22–26.

[30] Garud N Iyengar. "Robust dynamic programming". In: *Mathematics of Operations Research* 30.2 (2005), pp. 257–280.

[31] Nick Jakobi. “Evolutionary robotics and the radical envelope-of-noise hypothesis”. In: *Adaptive behavior* 6.2 (1997), pp. 325–368.

[32] Minqi Jiang et al. “Replay-guided adversarial environment design”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 1884–1897.

[33] Sarah Keren et al. “Efficient Heuristic Search for Optimal Environment Redesign”. In: *Proceedings of the International Conference on Automated Planning and Scheduling.* Vol. 29. 1. 2019, pp. 246–254.

[34] Sarah Keren et al. “Equi-reward utility maximizing design in stochastic environments”. In: *HSDIP 2017* (2017), p. 19.

[35] Rawal Khirodkar, Donghyun Yoo, and Kris M Kitani. “VADRA: Visual adversarial domain randomization and augmentation”. In: *arXiv preprint arXiv:1812.00491* (2018).

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.

[37] Tze Leung Lai and Herbert Robbins. “Asymptotically efficient adaptive allocation rules”. In: *Advances in applied mathematics* 6.1 (1985), pp. 4–22.

[38] Joel Z Leibo et al. “Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research”. In: *arXiv preprint arXiv:1903.00742* (2019).

[39] Zachary C Lipton. “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.” In: *Queue* 16.3 (2018), pp. 31–57.

[40] Tambet Matiisen et al. “Teacher-student curriculum learning”. In: *IEEE transactions on neural networks and learning systems* (2019).

[41] Christopher Mattson, Reamer L Bushardt, and Anthony R Artino Jr. *When a measure becomes a target, it ceases to be a good measure*. 2021.

[42] Eric Mazumdar, Lillian J Ratliff, and S Shankar Sastry. “On Gradient-Based Learning in Continuous Games”. In: *SIAM Journal on Mathematics of Data Science* 2.1 (2020), pp. 103–131.

[43] Ninareh Mehrabi et al. “A survey on bias and fairness in machine learning”. In: *ACM Computing Surveys (CSUR)* 54.6 (2021), pp. 1–35.

[44] Bhairav Mehta et al. “Active domain randomization”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 1162–1176.

[45] Smitha Milli et al. “Should robots be obedient?” In: *arXiv preprint arXiv:1705.09990* (2017).

[46] John Milnor. “Games Against Nature”. In: *Decision Processes* (1954), pp. 49–59.

[47] Matej Moravčık et al. "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker". In: *Science* 356.6337 (2017), pp. 508–513.

[48] Jun Morimoto and Kenji Doya. "Robust reinforcement learning". In: *Advances in neural information processing systems*. 2001, pp. 1061–1067.

[49] John F Nash Jr. "Equilibrium points in n-person games". In: *Proceedings of the national academy of sciences* 36.1 (1950), pp. 48–49.

[50] Arnab Nilim and Laurent El Ghaoui. "Robust control of Markov decision processes with uncertain transition matrices". In: *Operations Research* 53.5 (2005), pp. 780–798.

[51] Jack Parker-Holder et al. "Evolving Curricula with Regret-Based Environment Design". In: *arXiv preprint arXiv:2203.01302* (2022).

[52] Judea Pearl. *Causality*. Cambridge university press, 2009.

[53] Martin Peterson. *An introduction to decision theory*. Cambridge University Press, 2017.

[54] Lerrel Pinto, James Davidson, and Abhinav Gupta. "Supervision via competition: Robot adversaries for learning tasks". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 1601–1608.

[55] Aditya Ramesh et al. "Hierarchical text-conditional image generation with clip latents". In: *arXiv preprint arXiv:2204.06125* (2022).

[56] Sid Reddy, Anca Dragan, and Sergey Levine. "Pragmatic Image Compression for Human-in-the-Loop Decision-Making". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 26499–26510.

[57] Sid Reddy, Anca Dragan, and Sergey Levine. "Where do you think you're going?: Inferring beliefs about dynamics from behavior". In: *Advances in Neural Information Processing Systems* 31 (2018).

[58] Siddharth Reddy, Sergey Levine, and Anca D Dragan. "First Contact: Unsupervised Human-Machine Co-Adaptation via Mutual Information Maximization". In: *arXiv preprint arXiv:2205.12381* (2022).

[59] Kevin Regan and Craig Boutilier. "Regret-based reward elicitation for Markov decision processes". In: *arXiv preprint arXiv:1205.2619* (2012).

[60] Kevin Regan and Craig Boutilier. "Robust online optimization of reward-uncertain MDPs". In: *Twenty-Second International Joint Conference on Artificial Intelligence*. Citeseer. 2011.

[61] Herbert Robbins. "Some aspects of the sequential design of experiments". In: *Bulletin of the American Mathematical Society* 58.5 (1952), pp. 527–535.

[62] Fereshteh Sadeghi and Sergey Levine. "Cad2rl: Real single-image flight without a single real image". In: *arXiv preprint arXiv:1611.04201* (2016).

[63] Leonard J Savage. "The theory of statistical decision". In: *Journal of the American Statistical association* 46.253 (1951), pp. 55–67.

[64] Robert R Schaller. “Moore’s law: past, present and future”. In: *IEEE spectrum* 34.6 (1997), pp. 52–59.

[65] Jürgen Schmidhuber. “Formal theory of creativity, fun, and intrinsic motivation (1990–2010)”. In: *IEEE Transactions on Autonomous Mental Development* 2.3 (2010), pp. 230–247.

[66] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[67] Pierre Sermanet et al. “Time-contrastive networks: Self-supervised learning from video”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1134–1141.

[68] John Shalf. “The future of computing beyond Moore’s Law”. In: *Philosophical Transactions of the Royal Society A* 378.2166 (2020), p. 20190061.

[69] Claude Elwood Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[70] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[71] David Silver et al. “Mastering the game of go without human knowledge”. In: *nature* 550.7676 (2017), pp. 354–359.

[72] Satinder Singh, Richard L Lewis, and Andrew G Barto. “Where do rewards come from”. In: *Proceedings of the annual conference of the cognitive science society*. Cognitive Science Society. 2009, pp. 2601–2606.

[73] Jean-Jacques E Slotine and Weiping Li. “On the adaptive control of robot manipulators”. In: *The international journal of robotics research* 6.3 (1987), pp. 49–59.

[74] Sainbayar Sukhbaatar et al. “Intrinsic motivation and automatic curricula via asymmetric self-play”. In: *arXiv preprint arXiv:1703.05407* (2017).

[75] Richard Sutton. “The bitter lesson”. In: *Incomplete Ideas (blog)* 13 (2019), p. 12.

[76] Alexander Szalay and Jim Gray. “Science in an exponential world”. In: *Nature* 440.7083 (2006), pp. 413–414.

[77] Aviv Tamar, Shie Mannor, and Huan Xu. “Scaling up robust MDPs using function approximation”. In: *International Conference on Machine Learning*. 2014, pp. 181–189.

[78] Jie Tan et al. “Sim-to-real: Learning agile locomotion for quadruped robots”. In: *arXiv preprint arXiv:1804.10332* (2018).

[79] Josh Tobin et al. “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[80] Emanuel Todorov, Tom Erez, and Yuval Tassa. “Mujoco: A physics engine for model-based control”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033.

[81] Eugene Vinitsky et al. “D-MALT: Diverse Multi-Adversarial Learning for Transfer 003”. In: *Submission to ICML 2020)*.

[82] Abraham Wald. “Statistical decision functions.” In: (1950).

[83] Rui Wang et al. “Enhanced POET: Open-Ended Reinforcement Learning through Unbounded Invention of Learning Challenges and their Solutions”. In: *arXiv preprint arXiv:2003.08536* (2020).

[84] Rui Wang et al. “Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions”. In: *arXiv preprint arXiv:1901.01753* (2019).

[85] Chang Ye et al. “Rotation, Translation, and Cropping for Zero-Shot Generalization”. In: *arXiv preprint arXiv:2001.09908* (2020).

[86] Wenhao Yu et al. “Preparing for the unknown: Learning a universal policy with online system identification”. In: *arXiv preprint arXiv:1702.02453* (2017).

[87] Haoqi Zhang, Yiling Chen, and David C Parkes. “A general approach to environment design with one agent”. In: (2009).

[88] Haoqi Zhang and David C Parkes. “Value-Based Policy Teaching with Active Indirect Elicitation.” In: *AAAI*. Vol. 8. 2008, pp. 208–214.

[89] Kemin Zhou, John Comstock Doyle, Keith Glover, et al. *Robust and optimal control*. Vol. 40. Prentice hall New Jersey, 1996.

Appendix A

Generality of UED

In Section 4.2 we defined UPODMPS and UED and showed a selection of natural approaches to UED and corrisponding decision rules. However, the connection between UED and decisions under ignorance is much broader than these few decision rules. In fact, they can be seen as nearly identical problems.

To see the connection to decisions under ignorance, it is important to notice that any decision rule can be thought of as an ordering over policies, ranking the policies that it chooses higher than other policies. We will want to define a condition on this ordering, to do this we will first define what it means for one policy to *totally dominate* another.

Definition 8. *A policy, π_A, is* totally dominated *by some policy, π_B if for every pair of parameterizations $\vec{\theta}_A, \vec{\theta}_B$ $U^{\mathcal{M}_{\vec{\theta}_A}}(\pi_A) < U^{\mathcal{M}_{\vec{\theta}_B}}(\pi_B)$.*

Thus if π_A totally dominates π_B, it is reasonable to assume that π_A is better, since the best outcome we could hope for from policy π_B is still worse than the worst outcome we fear from policy π_A. Thus we would hope that our decision rule would not prefer π_B to π_A. If a decision rule respects this property we will say that it *respects total domination* or formally:

Definition 9. *We will say that an ordering $\prec$* respects total domination *iff $\pi_A \prec \pi_B$ whenever π_B totally dominates π_A.*

This is a very weak condition, but it is already enough to allow us to provide a characterization of all such orderings over policies in terms of policy-conditioned distributions over parameters $\vec{\theta}$, which we will notate Λ as in Section 4.2. Specifically, any ordering which respects total domination can be written as maximizing the expected value with respect to a *policy-conditioned value function*, thus every reasonable way of ranking policies can be described in the UED framework. For example, this implies that there is an environment policy which represents the strategy of minimax regret. We explicitly construct one such environment policy in Appendix B.

To make this formal, the *policy-conditioned value function*, $V^{\mathcal{M}_\Lambda}(\pi)$, is defined to be the expected value a policy will receive in the policy-conditioned distribution of environments

$\mathcal{M}_{\Lambda(\pi)}$, or formally:

$$V^{\mathcal{M}_\Lambda}(\pi) = \mathop{\mathbb{E}}_{\vec{\theta}\sim\Lambda(\pi)}[U^{\vec{\theta}}(\pi)] \tag{A.1}$$

.

The policy-conditioned value function is like the normal value function, but computed over the distribution of environments defined by the UPOMDP and environment policy. This of course implies an ordering over policies, defined in the natural way.

Definition 10. *We will say that* Λ *prefers* π_B *to* π_A *notated* $\pi_A \prec^\Lambda \pi_B$ *if* $V^{\mathcal{M}_\Lambda}(\pi_A) < V^{\mathcal{M}_\Lambda}(\pi_B)$

Finally, this allows us to state the main theorem of this Section.

Theorem 11. *Given an order over deterministic policies in a finite UPOMDP,* $\prec$*, there exits an environment policy* $\Lambda : \Pi \to \mathbf{\Delta}(\Theta^T)$ *such that* $\prec$ *is equivalent to* $\prec^\Lambda$ *iff it respects total domination and it ranks policies with an equal and a deterministic outcome as equal.*

Proof. Suppose you have some order of policies in a finite UPOMDP, $\prec$, which respects total domination and ranks policies equal if they have an equal and deterministic outcome. Our goal is to construct a function Λ such that $\pi_A \prec^\Lambda \pi_B$ iff $\pi_A \prec \pi_B$.

When constructing Λ notice that we can chose $\Lambda(\pi)$ independently for each policy π and that we can choose the resulting value for $V^{\mathcal{M}_\Lambda}(\pi)$ to lie anywhere within the range $[\min_{\vec{\theta}\in\Theta^T}\{U^{\vec{\theta}}(\pi)\}, \max_{\vec{\theta}\in\Theta^T}\{U^{\vec{\theta}}(\pi)\}]$. Since the number of deterministic policies in a finite POMDP is finite, we can build Λ inductively by taking the lowest ranked policy π in terms of $\prec$ for which Λ has not yet been defined and choosing the value for $V^{\mathcal{M}_\Lambda}(\pi)$ appropriately.

For the lowest ranked policy π_B for which $\Lambda(\pi_B)$ has not yet been defined we set $\Lambda(\pi_B)$ such that $V^{\mathcal{M}_\Lambda}(\pi_B)$ greater than $\Lambda(\pi_A)$ for all $\pi_A \prec \pi_B$ and is lower than the minimum possible value of any π_C such that $\pi_B \prec \pi_C$, if such a setting is possible. That is, we choose $\Lambda(\pi_B)$ to satisfy for all $\pi_A \prec \pi_B \prec \pi_C$:

$$\Lambda(\pi_A) < V^{\mathcal{M}_\Lambda}(\pi_B) < \max_{\vec{\theta}\in\Theta^T}\{U^{\vec{\theta}}(\pi_C)\} \tag{A.2}$$

Intuitively this ensures that $V^{\mathcal{M}_\Lambda}(\pi_B)$ is high enough to be above all π_A lower than π_B and low enough such that all future π_C can still be assigned an appropriate value.

Finally, we will show that it is possible to set $\Lambda(\pi_B)$ to satisfy these conditions in Equation A.2. By our inductive hypothesis, we know that $\Lambda(\pi_A) < \max_{\vec{\theta}\in\Theta^T}\{U^{\vec{\theta}}(\pi_B)\}$ and $\Lambda(\pi_A) < \max_{\vec{\theta}\in\Theta^T}\{U^{\vec{\theta}}(\pi_C)\}$. Since $\prec$ respects total domination, we know that $\min_{\vec{\theta}\in\Theta^T}\{U^{\vec{\theta}}(\pi_B)\} \leq \max_{\vec{\theta}\in\Theta^T}\{U^{\vec{\theta}}(\pi_C)\}$ for all $\pi_B \leq \pi_C$. Since there are a finite number of π_C we can set $V^{\mathcal{M}_\Lambda}(\pi_B)$ to be the average of the smallest value for $\max_{\vec{\theta}\in\Theta^T}\{U^{\vec{\theta}}(\pi_C)\}$ and the largest value for $\Lambda(\pi_A)$ for any π_C and π_A satisfying $\pi_A \prec \pi_B \prec \pi_C$.

The other direction, can be checked directly. If π_A is totally dominated by π_B, then:

$$V^{\mathcal{M}_\Lambda}(\pi_A) = \mathop{\mathbb{E}}_{\vec{\theta}\sim\Lambda(\pi_A)}[U^{\vec{\theta}}(\pi_A)] < \mathop{\mathbb{E}}_{\vec{\theta}\sim\Lambda(\pi_B)}[U^{\vec{\theta}}(\pi_B)] = V^{\mathcal{M}_\Lambda}(\pi_B)$$

Thus if $\prec$ can be represented with an appropriate choice of Λ then it respects total domination and it ranks policies with an equal and a deterministic outcome equal. $\square$

Thus, the set of decision rules which respect total domination are exactly those which can be written by some concrete environment policy, and thus any approach to making decisions under uncertainty which has this property can be thought of as a problem of unsupervised environment design and vice-versa. To the best of our knowledge there is no seriously considered approach to making decisions under uncertainty which does not satisfy this property.

Appendix B

Defining an Environment Policy Corresponding to Minimax Regret

In this section we will be deriving a function $\Lambda(\pi)$ which corresponds to regret minimization. We will be assuming a finite MDP with a bounded reward function. Explicitly, we will be deriving a Λ such that minimax regret is the decision rule which maximizes the corresponding *policy-conditioned value function* defined in Appendix A:

$$V^{\mathcal{M}_\Lambda}(\pi) = \mathop{\mathbb{E}}_{\vec{\theta}\sim\Lambda(\pi)}[U^{\vec{\theta}}(\pi)]$$

In this context, we will define regret to be:

$$\textsc{Regret}(\pi, \vec{\theta}) = \max_{\pi^B \in \Pi}\{U^{\mathcal{M}_{\vec{\theta}}}(\pi^B) - U^{\mathcal{M}_{\vec{\theta}}}(\pi)\}$$

Thus the set of $\textsc{MinimaxRegret}$ strategies can be defined naturally as the ones which achive the minimum worst case regret:

$$\textsc{MinimaxRegret} = \operatorname*{argmin}_{\pi\in\Pi}\{\max_{\vec{\theta}\in\Theta^T}\{\textsc{Regret}(\pi, \vec{\theta})\}\}$$

We want to find a function Λ for which the set of optimal policies which maximize value with respect to Λ is the same as the set of $\textsc{MinimaxRegret}$ strategies. That is, we want to find Λ such that:

$$\textsc{MinimaxRegret} = \operatorname*{argmax}_{\pi\in\Pi}\{V^{\mathcal{M}_\Lambda}(\pi)\}$$

To do this it is useful to first introduce the concept of *weak total domination*, which is the natural weakening of the concept of total domination introduced in Appendix A. A policy is said to be weakly totally dominated by a policy π_B if the maximum outcome that can be achieved by π_A is equal to the minimum outcome that can be achieved by π_B, or formally:

Definition 12. *A policy, π_A, is* weakly totally dominated *by some policy, π_B if for every pair of parameterizations $\vec{\theta}_A, \vec{\theta}_B$ $U^{\mathcal{M}_{\vec{\theta}_A}}(\pi_A) \leq U^{\mathcal{M}_{\vec{\theta}_B}}(\pi_B)$.*

The concept of weak total domination is only needed here to clarify what happens in the special case that there are policies that are weakly totally dominated but not totally dominated. These policies may or may not be minimax regret optimal policies, and thus have to be treated with more care. To get a general sense for how the proof works you may assume that there are no policies which are weakly totally dominated but not totally dominated and skip the sections of the proof marked as a special case: the second paragraph of the proof for Lemma 13 and the second paragraph of the proof for Theorem 14.

We can use this concept to help define a normalization function $D : \Pi \to \mathbf{\Delta}(\Theta^T)$ which has the property that if there are two policies π_A, π_B such that neither is totally dominated by the other, then they evaluate the same value on the distribution of environments. We will use D as a basis for creating Λ by shifting probability mass towards or away from this distribution in the right proportion. In general there are many such normalization functions, but we will simply show that one of these exists.

Lemma 13. *There exists a function $D : \Pi \to \mathbf{\Delta}(\Theta^T)$ such that $\Lambda(\pi)$ has support at least $s > 0$ on the highest-valued regret-maximizing parameterization $\overline{\theta}_\pi$ for all π, that for all π_A, π_B such that neither is weakly totally dominated by any other policy, $V^{\mathcal{M}_D}(\pi_A) = V^{\mathcal{M}_D}(\pi_B)$ and $V^{\mathcal{M}_D}(\pi_A) > V^{\mathcal{M}_D}(\pi_B)$ when π_B is totally dominated and π_A is not. If the policy π is weakly dominated but not totally dominated we choose D to put full support on the highest-valued regret-maximizing parameterization, $D(\pi) = \overline{\theta}_\pi$.*

Proof. We will first define D for the set of policies which are not totally dominated, which we will call X. Note that by the definition of not being totally dominated, there is a constant C which is between the worst-case and best-case values of all of the policies in X.

Special Case: If there is only one choice for C and if that does have support over $\overline{\theta}_\pi$ for each π which is not totally dominated, then we know that there is a weakly totally dominated policy which is not totally dominated. In this case we choose a C which is between the best-case and the worst-case for the not weakly totally dominated policies. Thus for each π which is not weakly dominated we can find a distribution $\vec{\theta}$ such that $U^{\vec{\theta}}(\pi) = C$. If C is not the maximum or minimum achievable value for π then we will chose $D(\pi)$ to have expected value C and and have support over all Θ^T.

Otherwise, we can choose $D(\pi) = \overline{\theta}_\pi$. For other $\pi \notin X$, we can let $D(\pi) = U(\Theta^T)$, which achieves value less than C since all outcomes for π have utility less than C. Thus, by construction D satisfies the desired conditions. □

Given such a function D, we will construct a Λ which works by shifting probability towards or away from the environment parameters that maximize the regret for the given agent, in the right proportions to achieve the desired result. We claim that the following definition works:

$$\Lambda(\pi) = \{\overline{\theta}_\pi : \frac{c_\pi}{v_\pi}, \tilde{D}_\pi : \text{otherwise}\}$$

Where the bracket notation defines a mixture distribution $\{x : P, y : Q\}(\theta) = xP(\theta) + yQ(\theta)$ and where $\tilde{D}_\pi = D(\pi)$ is a baseline distribution satisfying the conditions of Lemma 13,

$\overline{\theta}_\pi$ is the trajectory which maximizes regret of π, and v_π is the value above the baseline distribution that π achieves on that trajectory, and c_π is the negative of the worst-case regret of π, normalized so that $\frac{c_\pi}{v_\pi}$ is between $-s$ and 1. If $\frac{c_\pi}{v_\pi}$ is negative then we will interpret probability mass of $\frac{c_\pi}{v_\pi}$ on $\overline{\theta}_\pi$ in $\tilde{D}_\pi$ being redistributed proportionally in $\tilde{D}_\pi$ as is implied by the equation. Note that if $v_\pi = 0$ then $\tilde{D}_\pi = \overline{\theta}_\pi$ by our construction, so $\frac{c_\pi}{v_\pi}$ can be interpreted to be anything in $[0, 1]$ without effecting the distribution $\Lambda(\pi)$.

Theorem 14. *For* Λ *as defined above:*

$$\operatorname*{argmax}_{\pi\in\Pi}\{\mathop{\mathbb{E}}_{\vec{\theta}\sim\Lambda(\pi)}[U^{\vec{\theta}}(\pi)]\} = \textsc{MinimaxRegret}$$

Proof. By the construction of $D(\pi)$ by Lemma 13, we will let $C = \mathop{\mathbb{E}}_{\vec{\theta}\sim\tilde{D}_\pi}[U^{\vec{\theta}}(\pi)]$, which is a constant independent of π by construction for all of the policies which are not weakly totally dominated. If there are no policies which are weakly totally dominated but not totally dominated the next paragraph can be skipped, otherwise it handles the special case by showing that the conditions we ensured for the non-totally dominated policies hold for these weakly totally dominated policies if and only if they are minimax regret optimal policies.

Special Case: For the π which are weakly totally dominated, but not totally dominated, we will show that $\mathop{\mathbb{E}}_{\vec{\theta}\sim\tilde{D}_\pi}[U^{\vec{\theta}}(\pi)] = C$ if $\pi \in \textsc{MinimaxRegret}$ and $\mathop{\mathbb{E}}_{\vec{\theta}\sim\tilde{D}_\pi}[U^{\vec{\theta}}(\pi)] < C$ if $\pi \notin \textsc{MinimaxRegret}$. Let M be the maximum value achieved by any policy. Then if $\pi \in \textsc{MinimaxRegret}$ and the maximum value it achieves is C' then note that any policy that weakly dominates π, say π', can achieve no more than $M - C'$ regret. Thus if π is a minimax optimal strategy, it must achieve exactly $M - C'$ regret and both π and π' must have a regret maximizing outcome at the same value $M - C'$, which is the maximum value outcome for π and the minimum value outcome for π'. Thus $C = C'$ since both π and π' must have support on the an outcome of value C' and $\mathop{\mathbb{E}}_{\vec{\theta}\sim\tilde{D}_\pi}[U^{\vec{\theta}}(\pi)] = C$. Otherwise $\pi \notin \textsc{MinimaxRegret}$ so it achieves more than $M - C'$ regret and $\mathop{\mathbb{E}}_{\vec{\theta}\sim\tilde{D}_\pi}[U^{\vec{\theta}}(\pi)] < C$.

With this case out of the way, we can use the definition of Λ and simplify to show the desired result.

$$\begin{aligned}
\underset{\pi\in\Pi}{\operatorname{argmax}}\{\underset{\vec{\theta}\sim\Lambda(\pi)}{\mathbb{E}}[U^{\vec{\theta}}(\pi)]\} &= \underset{\pi\in\Pi}{\operatorname{argmax}}\{\frac{U^{\bar{\theta}_\pi}(\pi)c_\pi}{v_\pi} + C(1-\frac{c_\pi}{v_\pi})\} && \text{(B.1)}\\
&= \underset{\pi\in\Pi}{\operatorname{argmax}}\{\frac{v_\pi c_\pi + Cc_\pi}{v_\pi} + (\frac{Cv_\pi - Cc_\pi}{v_\pi})\} && \text{(B.2)}\\
&= \underset{\pi\in\Pi}{\operatorname{argmax}}\{\frac{v_\pi c_\pi + Cv_\pi}{v_\pi}\} && \text{(B.3)}\\
&= \underset{\pi\in\Pi}{\operatorname{argmax}}\{c_\pi + C\} && \text{(B.4)}\\
&= \underset{\pi\in\Pi}{\operatorname{argmax}}\{c_\pi\} && \text{(B.5)}\\
&= \underset{\pi\in\Pi}{\operatorname{argmax}}\{\min_{\vec{\theta}\in\Theta^T}\{-\text{Regret}(\pi,\vec{\theta})\}\} && \text{(B.6)}\\
&= \underset{\pi\in\Pi}{\operatorname{argmin}}\{\max_{\vec{\theta}\in\Theta^T}\{\text{Regret}(\pi,\vec{\theta})\}\} && \text{(B.7)}
\end{aligned}$$

□

Appendix C

Minimax Regret Always Succeeds when There is a Clear Notion of Success

In this section we will show that when there is a sufficiently strong notion of success and failure, and there is a policy which can ensure success, minimax regret will choose a successful strategy. Moreover, we will show that neither pure randomization, nor maximin has this property.

Theorem 6. *Suppose that all achievable rewards fall into one of two class of outcomes labeled* ***SUCCESS*** *giving rewards in* $[\boldsymbol{S}_{min}, \boldsymbol{S}_{max}]$ *and* ***FAILURE*** *giving rewards in* $[\boldsymbol{F}_{min}, \boldsymbol{F}_{max}]$*, such that* $\boldsymbol{F}_{min} \leq \boldsymbol{F}_{max} < \boldsymbol{S}_{min} \leq \boldsymbol{S}_{max}$*. In addition assume that the range of possible rewards in either class is smaller than the difference between the classes so we have* $\boldsymbol{S}_{max} - \boldsymbol{S}_{min} < \boldsymbol{S}_{min} - \boldsymbol{F}_{max}$ *and* $\boldsymbol{F}_{max} - \boldsymbol{F}_{min} < \boldsymbol{S}_{min} - \boldsymbol{F}_{max}$*. Further suppose that there is a policy* π *which succeeds on any* $\vec{\theta}$ *whenever success is possible. Then minimax regret will choose a policy which has that property.*

Further suppose that there is a policy π *which succeeds on any* $\vec{\theta}$ *whenever any policy succeeds on* $\vec{\theta}$*. Then minimax regret will choose a policy which has that property.*

Proof. Let $C = \mathbf{S}_{min} - \mathbf{F}_{max}$ and let π^* be one of the policies that succeeds on any $\vec{\theta}$ whenever success is possible. By assumption $\mathbf{S}_{max} - \mathbf{S}_{min} < C$ and $\mathbf{F}_{max} - \mathbf{F}_{min} < C$. Since π^* succeeds whenever success is possible we have:

$$\max_{\vec{\theta} \in \Theta^T} \{\textsc{Regret}(\pi^*, \vec{\theta})\} < C$$

Thus any strategy which achieves minimax regret must have regret less than C, since it must do at least as well as π^*. Suppose $\pi \in \textsc{MinimaxRegret}$, then if π does not solve some solvable $\vec{\theta}$ then $\textsc{Regret}(\pi^*, \vec{\theta}) > C$. Thus every minimax regret policy succeeds whenever that is possible. $\square$

Though minimax regret has this property, the other decision rules considered do not. For example, consider the task described by Table C.1a, where an entry in position (i, j) shows the payoff $U^{\theta_j}(\pi_i)$ obtained for the policy π_i in row i, in the environment parameterized by θ_j in column j. If you choose π_B you can ensure success whenever any policy can ensure success, but maximin will choose π_A, as it only cares about the worst case outcome.

	θ_1	θ_2
π_A	0	0
π_B	100	-1

(a)

	θ_1	θ_2	θ_3	θ_4	θ_5
π_A	75	75	75	75	75
π_B	0	100	100	100	100
π_C	100	0	100	100	100
π_D	100	100	0	100	100
π_E	100	100	100	0	100
π_F	100	100	100	100	0

(b)

Table C.1: In these setting, **SUCCESS** is scoring in the range $[75, 100]$ and **FAILURE** is scoring in the range $[-1, 0]$

Choosing the policy that maximizes the expected return fails in the game described by Table C.1b. Under a uniform distribution all of $\pi_B, \pi_C, \pi_D, \pi_E, \pi_F$ give an expected value of 80, while π_A gives an expected value of 75. Here maximizing expected value on a uniform distribution will choose one of $\pi_B, \pi_C, \pi_D, \pi_E, \pi_F$ while π_A guarantees success. Moreover, this holds for every possible distribution over parameters, since π_A will always give an expected value of 75 and on average $\pi_B, \pi_C, \pi_D, \pi_E, \pi_F$ will give an expected value of 80, so at least one of them gives above 80.

Appendix D

Nash solutions to PAIRED

In this section we show how PAIRED works when coordination is not achieved. The following proof shows that if the antagonist, protagonist, and adversary find a Nash Equilibrium, then the protagonist performs better than or equal to the antagonist in every parameterization.

Theorem 15. *Let $(\pi^P, \pi^A, \vec{\theta})$ be in Nash equilibria. Then for all $\vec{\theta}'$: $U^{\vec{\theta}'}(\pi_P) \geq U^{\vec{\theta}'}(\pi_A)$.*

Proof. Let $(\pi^P, \pi^A, \vec{\theta})$ be in Nash equilibria. Then regardless of $\pi^A, \vec{\theta}$, the protagonist always has the choice to play $\pi^P = \pi^A$, so $\textsc{Regret}(\pi^P, \pi^A, \vec{\theta}) \leq 0$. Since, the adversary chooses $\vec{\theta}$ over any other $\vec{\theta}'$ we know that $\textsc{Regret}(\pi^P, \pi^A, \vec{\theta}') \leq 0$ must hold for all $\vec{\theta}'$. Thus by the definition of Regret $U^{\vec{\theta}'}(\pi_P) \geq U^{\vec{\theta}'}(\pi_A)$. □

This shows that with a capable antagonist the protagonist would learn the minimax regret policy, even without the adversary and the antagonist coordinating, and suggests that methods which strengthen the antagonist could serve to improve the protagonist.

Appendix E

Additional experiments

E.0.1 Alternative methods for computing regret

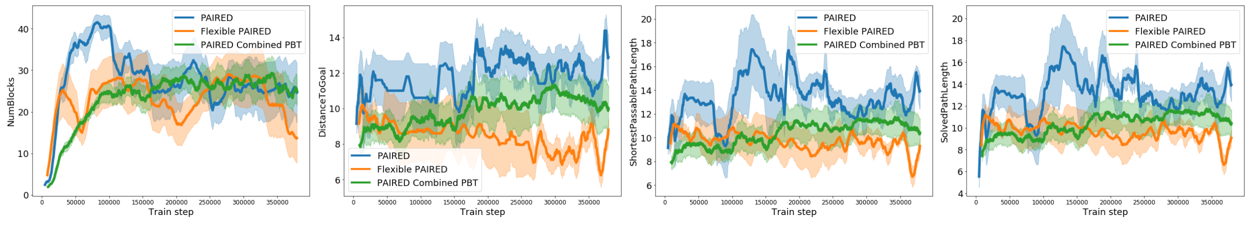
Given the arguments in Appendix D, we experimented with additional methods for approximating the regret, which attempt to improve the antagonist. We hypothesized that using a fixed agent as the antagonist could potentially lead to optimization problems in practice; if the antagonist became stuck in a local minimum and stopped learning, it could limit the complexity of the environments the adversary could generate. Therefore, we developed an alternative approach using population-based training (PBT), where for each environment designed by the adversary, each agent in the population collects a single trajectory in the environment. Then, the regret is computed using the difference between the maximum performing agent in the population, and the mean of the population. Assume there is a population of K agents, and i and j index agents within the population. Then:

$$\textsc{Regret}_{pop} = \max_i U(\tau^i) - \frac{1}{K}\sum_{j=1}^{K}[U(\tau^j)] \tag{E.1}$$

We also used a population of adversaries, where for each episode, we randomly select an adversary to generate the environment, then test all agents within that environment. We call this approach *PAIRED Combined PBT*.

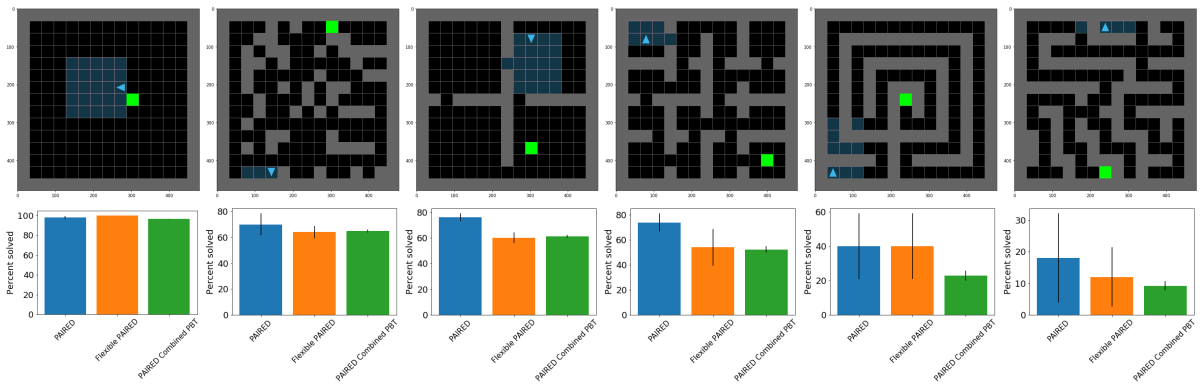
Since using PBT can be expensive, we also investigated using a similar approach in the case of a population of $K = 2$ agents, and a single adversary. This is analogous to a version of PAIRED where there is no fixed antagonist, but the antagonist is flexibly chosen to be the currently best-performing agent. We call this approach *Flexible PAIRED*.

Figure E.1 plots the complexity results of these two approaches. We find that they both perform worse than the proposed version of PAIRED. Figure E.2 shows the transfer performance of both methods against the original PAIRED. Both methods achieve reasonable transfer performance, retaining the ability to solve complex test tasks like the labyrinth and maze (when domain randomization and minimax training cannot). However, we find that the original method for computing the regret outperforms both approaches. We note that both



(a) Number of blocks (b) Distance to goal (c) Passable path length (d) Solved path length

Figure E.1: Comparison of alternative methods for computing the regret in terms of the statistics of generated environments (a-c), and the length of mazes that agents are able to solve (d). Neither the combined population or flexible approach (which both break the coordination assumption in Theorem 2) show improved complexity. Statistics of generated and solved environments measured over five random seeds; error bars are a 95% CI.



(a) Empty (b) 50 Blocks (c) 4 Rooms (d) 16 Rooms (e) Labyrinth (f) Maze

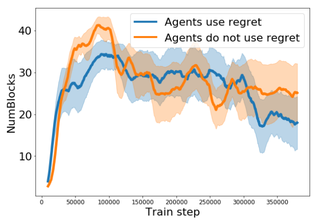
Figure E.2: Comparison of transfer performance for alternative methods for computing the regret. While both alternatives still retain the ability to solve complex tasks, achieving superior performance to minimax and domain randomization, their performance is not as good as the proposed method for computing the regret.

the combined population and flexible PAIRED approaches do not enable the adversary and antagonist to coordinate (since the antagonist does not remain fixed). Coordination between the antagonist and adversary is an assumption in Theorem 2 of Section 4.3, which shows that if the agents reach a Nash equilibrium, the protagonist will be playing the minimum regret policy. These empirical results appear to indicate that when coordination between the adversary and antagonist is no longer possible, performance degrades. Future work should investigate whether these results hold across more domains.

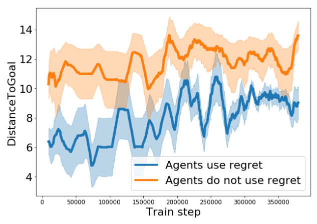
E.0.2 Should agents themselves optimize regret?

We experimented with whether the protagonist and antagonist should optimize for regret, or for the normal reward supplied by the environment (note that the environment-generating adversary always optimizes for the regret). For both the protagonist and antagonist, the regret is based on the difference between their reward for the current trajectory, and the max reward of the other agent received over several trajectories played in the current environment. Let the current agent be A, and the other agent be O. Then agent A should receive reward $R^A = U(\tau^A) - \max_{\tau^O} U(\tau^O)$ for episode τ^A. However, this reward is likely to be negative. If given at the end of the trajectory, it would essentially punish the agent for reaching the goal, making learning difficult. Therefore we instead compute a per-timestep penalty of $\frac{\max_{\tau^O} U(\tau^O)}{T}$, where T is the maximum length of an episode. This is subtracted post-hoc from each agent's reward at every step during each episode, after the episode has been collected but before the agent is trained on it. When the agent reaches the goal, it receives the normal Minigrid reward for successfully navigating to the goal, which is $R = 1 - 0.9 * (M/T)$, where M is the timestep on which the agent found the goal.

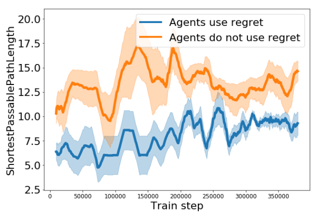
Figure E.3 shows the complexity results for the best performing hyperparameters in which the protagonist and antagonist optimized the regret according to the formula above, or whether they simply learned according to the environment reward. Note that in both cases, the environment-generating adversary optimizes the regret of the protagonist with respect to the antagonist. As is evident in Figure E.3, training agents on the environment reward itself, rather than the regret, appears to be more effective for learning complex behavior. We hypothesize this is because the regret is very noisy. The performance of the other agent is stochastic, and variations in the other agent's reward are outside of the agent's control. Further, agents do not receive observations about the other agent, and cannot use them to determine what is causing the reward to vary. However, we note that optimizing for the regret can provide good transfer performance. The transfer plots in Figure 4.3 were created with an agent that optimized for regret, as we describe below. It is possible that as the other agent converges, the regret provides a more reliable signal indicating when the agent's performance is sub-optimal.



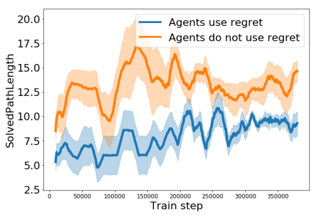
(a) Number of blocks



(b) Distance to goal



(c) Passable path length



(d) Solved path length

Figure E.3: Comparison of alternative methods for computing the regret in terms of the statistics of generated environments (a-c), and the length of mazes that agents are able to solve. Neither the combined population or flexible approach show improved complexity. Statistics of generated and solved environments measured over five random seeds; error bars are a 95% CI.

Appendix F

Experiment Details and Hyperparameters

F.0.1 Navigation Experiments

F.0.1.1 Agent parameterization

The protagonist and antagonist agents for both PAIRED and the baselines received a partially observed, $5 \times 5 \times 3$ view of the environment, as well as an integer ranging from $0 - 3$ representing the direction the agent is currently facing. The agents use a network architecture consisting of a single convolutional layer which connects to an LSTM, and then to two fully connected layers which connect to the policy outputs. A second network with identical parameters is used to estimate the value function. The agents use convolutional kernels of size 3 with 16 filters to input the view of the environment, an extra fully-connected layer of size 5 to process the direction and input it to the LSTM, an LSTM of size 256, and two fully connected layers of size 32 which connect to either the policy outputs or the value estimate. The best entropy regularization coefficient was found to be 0.0. All agents (including environment adversaries) are trained with PPO with a discount factor of 0.995, a learning rate of 0.0001, and 30 environment workers operating in parallel to collect a batch of episodes, which is used to complete one training update.

F.0.1.2 Adversary parameterization

The environments explored in this paper are a 15×15 tile discrete grid, with a border of walls around the edge. This means there are $13 \times 13 = 169$ free tiles for the adversary to use to place obstacles. We parameterized the adversary by giving it an action space of dimensionality 169, and each discrete action indicates the location of the next object to be placed. It plays a sequence of actions, such that on the first step it places the agent, on the second it places the goal, and for 50 steps afterwards it places a wall (obstacle). If the adversary places an object on top of a previously existing object, its action does nothing;

this allows it to place fewer than 50 obstacles. If it tries to place the goal on top of the agent, the goal will be placed randomly.

We also explored an alternative parameterization, in which the adversary had only 4 actions, which corresponded to placing the agent, goal, an obstacle, or nothing. It then took a sequence of 169 steps to place all objects, moving through the grid from top to bottom and left to right. If it chose to place the agent or goal when they had already been placed elsewhere in the map, they would be moved to the current location. This parameterization allows the adversary to place as many blocks as there are squares in the map. However, we found that adversaries trained with this parameterization drastically underperformed the alternative version used in the paper, scoring an average solved path length of ≈ 2, as opposed to ≈ 15. We hypothesize this is because when the adversary is randomly initialized, sampling from its random policy is more likely to produce impossible environments. This makes it impossible for the agents to learn, and cannot provide a regret signal to the adversary to allow it to improve. We suggest that when designing an adversary parameterization, it may be important to ensure that sampling from a random adversary policy can produce feasible environments.

The environment-constructing adversary's observations consist of a $15 \times 15 \times 3$ image of the state of the environment, an integer t representing the current timestep, and a random vector $z \sim \mathcal{N}(0, I), z \in \mathbb{R}^{50}$ to allow it to generate random mazes. Because the sequencing of actions is important, we experiment with using an RNN to parameterize the adversary as well, although we find it is not always necessary. The adversary architecture is similar to that of the agents; it consists of a single convolutional layer which connects to an LSTM, and then to two fully connected layers which connect to the policy outputs. Additional inputs such as t and z are connected directly to the LSTM layer. We use a second, identical network to estimate the value function. To find the best hyperparameters for PAIRED and the baselines, we perform a grid search over the number of convolution filters used by the adversary, the degree of entropy regularization, which of the two parameterizations to use, and whether or not to use an RNN, and finally, the number of steps the protagonist is given to find the goal (we reasoned that lowering the protagonist episode length relative to the antagonist length would make the regret a less noisy signal). These parameters are shared between PAIRED and the baseline Minimax adversary, and we sweep all of these parameters equally for both. For PAIRED, we also experiment with whether the protagonist and antagonist optimize regret, and with using a non-negative regret signal, *i.e.* $\textsc{Regret} = \max(0, \textsc{Regret})$, reasoning this could also lead to a less noisy reward signal.

For the complexity experiments, we chose the parameters that resulted in the highest solved path length in the last 20% of training steps. The best parameters for PAIRED were an adversary with 128 convolution filters, entropy regularization coefficient of 0.0, protagonist episode length of 250, non-negative regret, and the agents did not optimize regret. The best parameters for the minimax adversary were 256 convolution filters, entropy regularization of 0.0, and episode length of 250. For the Population-based-training (PBT) Minimax experiment, the best parameters were an adversary population of size 3, and an agent population of size 3. All adversaries used a convolution kernel size of 3, an LSTM size of 256, two fully connected layers of size 32 each, and a fully connected layer of size 10 that inputs the timestep and connects to the LSTM.

For the transfer experiments, we first limited the hyperparameters we searched based on those which produced the highest adversary reward, reasoning that these were experiments in which the optimization objective was achieved most effectively. We tested a limited number of hyperparameter settings on a set of validation environments, consisting of a different maze and labyrinth, mazes with 5 blocks and 40 blocks, and a nine rooms environment. We then tested these parameters on novel test environments to produce the scores in the paper. The best parameters for the PAIRED adversary were found to be 64 convolution filters, entropy regularization of 0.1, no RNN, non-negative regret, and having the agents themselves optimize regret. The best parameters for the minimax adversary in this experiment were found to be the same: 64 convolution filters, entropy regularization of 0.1, and no RNN.

F.0.2 Hopper Experiments

The hopper experiments in this paper are in the standard MuJoCo Todorov, Erez, and Tassa [80] simulator. The adversary is allowed to apply additional torques to the joints of the agent at a some proportion of the original agent's strength α. The torques that are applied at each time step are chosen by the adversary independent of the state of the environment, and in the PAIRED experiments, both the protagonist and the antagonist are given the same torques.

The adversaries observation consists of only of the the time step. Each policy is a DNN with two hidden layers each with width 32, *tanh* activation internally and a linear activation on the output layer. They were trained simultaneously using PPO Schulman et al. [66] and an schedule in the adversary strength parameter α is scaled from 0.1 to 1.0 over the course of 300 iterations, after which training continues at full strength. We pre-train agents without any adversary for 100 iterations.

Hyperparameter tuning was conducted over the learning rate values [5e-3, 5e-4, 5e-5] and the GAE lambda values $[0.5, 0.9]$. The minimax adversary worked best with a leaning rate of 5e-4 and a lambda value of 0.5; PAIRED worked best with a leaning rate of 5e-3 and a lambda value of 0.09. Otherwise we use the standard hyperparameters in Ray 0.8.