On the Semantics of Purpose Requirements in Privacy Policies

Michael Carl Tschantz **Anupam Datta**
Jeannette M. Wing

February 18, 2011
CMU-CS-11-102

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Privacy policies often place requirements on the purposes for which a governed entity may use personal information. For example, regulations, such as HIPAA, require that hospital employees use medical information for only certain purposes, such as treatment. Thus, using formal or automated methods for enforcing privacy policies requires a semantics of *purpose requirements* to determine whether an action is *for* a purpose or not. We provide such a semantics using a formalism based on *planning*. We model planning using a modified version of Markov Decision Processes, which exclude redundant actions for a formal definition of *redundant*. We use the model to formalize when a sequence of actions is *only for* or *not for* a purpose. This semantics enables us to provide an algorithm for automating auditing, and to describe formally and compare rigorously previous enforcement methods.

This research was supported by the US Army Research Office under grant numbers W911NF0910273 and DAAD-190210389. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

This manuscript was submitted to the 24th IEEE Computer Security Foundations Symposium.

Keywords: Privacy, Formal Methods

1 Introduction

Purpose is a key concept for privacy policies. For example, the European Union requires that [The95]:

> Member States shall provide that personal data must be [...] collected for specified, explicit and legitimate purposes and not further processed in a way incompatible with those purposes.

The United States also has laws placing purpose requirements on information in some domains such as HIPAA [Off03] for medical information and the Gramm-Leach-Bliley Act [Uni10] for financial records. These laws and best practices motivate organizations to discuss in their privacy policies the purposes for which they will use information.

Some privacy policies warn users that the policy provider may use certain information for certain purposes. For example, the privacy policy of a medical provider states, "We may disclose your [protected health information] for public health activities and purposes [...]" [Was03]. Such warnings do not constrain the behavior of the policy provider.

Other policies that prohibit using certain information for a purpose do constrain the behavior of the policy provider. Examples include the privacy policy of Yahoo! Email, which states that "Yahoo!'s practice is *not* to use the content of messages stored in your Yahoo! Mail account *for* marketing purposes" [Yah10b, emphasis added].

Some policies even limit the use of certain information to an explicit list of purposes. The privacy policy of The Bank of America states, "Employees are authorized to access Customer Information *for* business purposes *only*." [Ban05, emphasis added]. The HIPAA Privacy Rule [Off03] requires that covered entities (e.g., health care providers and business partners) only use or disclose protected health information about a patient with that patient's written authorization or:

> [...] for the following purposes or situations: (1) To the Individual [...]; (2) Treatment, Payment, and Health Care Operations; (3) Opportunity to Agree or Object; (4) Incident to an otherwise permitted use and disclosure; (5) Public Interest and Benefit Activities; and (6) Limited Data Set for the purposes of research, public health or health care operations.

These examples show that verifying that an organization obeys a privacy policy requires a semantics of *purpose requirements*. In particular, enforcement requires the ability to determine that the organization under scrutiny obeys at least two classes of purpose requirements. As shown in the example rule from Yahoo!, the first requirement is that the organization does *not* use certain sensitive information *for* a given purpose. The second, as the example rule from HIPAA shows, is that the organization uses certain sensitive information *only for* a given list of purposes. We call the first class of requirements *prohibitive* (not-for) and the second class *restrictive* (only-for). Each class requires determining whether the organization's behavior is *for* a purpose or not, but they differ in whether this indicates a violation or compliance, respectively.

For example, consider a physician accessing a medical record. Under the HIPAA Privacy Rule, the physician may access the record only for certain purposes such as treatment, research, and billing. Thus, for an auditor (either internal or external) to determine whether the physician has obeyed the Privacy Rule requires the auditor to determine the purposes for which the physician accessed the record. The auditor's ability to determine the purposes behind actions is limited since the auditor can only observe the behavior of the physician. As a physician may perform the exact

same actions for different purposes, the auditor can never be sure of the purposes behind an action. However, if the auditor determines that the record access could not have possibly been for any of the purposes allowed under the Privacy Rule, then the auditor knows that the physician violated the policy.

Manual enforcement of these privacy policies is labor intensive and error prone. Thus, to reduce costs and make their operations more trustworthy, organizations would like to automate the enforcement of the privacy policies governing their operations; tool support for this activity is beginning to emerge in the market. For example, Fair Warning offers automated services for the detection of privacy breaches in a hospital setting [Fai]. Meanwhile, previous research has purposed formal methods to enforce purpose requirements [AKSX02, BBL05, HA05, AF07, BL08, PGY08, JSNS09, NBL$^+$10, EKWB11].

However, each of these endeavors start by assuming that actions or sequences of actions are labeled with the purposes they are *for*. They avoid analyzing the meaning of *purpose* and provide no method of performing this labeling other than through intuition alone. The absence of a formal semantics to guide this determination has hampered the development of methods for ensuring policy compliance. Such a definition would provide insights into how to develop tools that identify suspicious accesses in need of detailed auditing and algorithms for determining which purposes an action could possibly be for. Such a definition would also show which enforcement approaches are most accurate. More fundamentally, such a definition could frame the scientific basis of a societal and legal understanding of purpose and of privacy policies that use the notion of purpose. Such a foundation can, for example, guide implementers as they codify in software an organization's interpretation of internal and government-imposed privacy policies.

1.1 Solution Approach

The goal of this work is to study the meaning of *purpose* in the context of enforcing privacy policies and propose formal definitions suitable for automating the enforcement of purpose requirements. Since post-hoc auditing provides the perspective often required to determine the purpose of an action, we focus on automated auditing. However, we believe our semantics is applicable to other formal methods and may also clarify informal reasoning.

We find that *planning* is central to the meaning of purpose. We see the role of planning in the definition of the sense of the word "purpose" most relevant to our work [OED89]:

> The object for which anything is done or made, or for which it exists; the result or effect intended or sought; end, aim.

Similarly, work on cognitive psychology calls purpose "the central determinant of behavior" [DKP96, p19]. If our auditors are concerned with rational auditees (the person or organization being audited), then we may assume the auditee uses a plan to determine what actions it will perform in its attempt to achieve its purposes. We (as have philosophers [Tay66]) conclude that if an auditee selects to perform an action a while planning to achieve the purpose p, then the auditee's action a is *for the purpose* p. In this paper, we make these notions formal.

1.2 Overview of Contributions

We first present an example that illustrates key factors in determining whether an action is for a purpose or not. We find that the auditor should model the auditee as an agent that interacts with

an *environment model*. The environment model shows how the actions the auditee can perform affect the state of the environment. It also models how well each state satisfies each purpose that the modeled auditee might possibly find motivating. Limiting consideration to one purpose, the environment model becomes a Markov Decision Process (MDP) where the degree of satisfaction of that purpose is the reward function of the MDP. If the auditee is motivated to act by only that purpose, then the auditee's actions must correspond to an optimal *plan* for this MDP and these actions are *for* that purpose. Additionally, we use a stricter definition of optimal than standard MDPs to reject redundant actions that neither decrease nor increase the total reward. We formalize this model in Section 3.

For example, consider a physician ordering a medical test and an auditor attempting to determine whether the physician could have ordered this test for the purpose of treatment (and is therefore in compliance with the HIPAA Privacy Rule). The auditor would examine an MDP modeling the physician's environment with the quality of treatment as the reward function to be optimized. If no optimal plans for this MDP involve ordering the test, then the auditor can conclude definitively that the physician did not order the test for treatment.

We make this auditing process formal in Section 4 where we discuss the ramifications of the auditor only observing the behaviors of the auditee and not the underlying planning process of the auditee that resulted in these behaviors. We show that in some circumstances, the auditor can still acquire enough information to determine that the auditee violated the privacy policy. To do so, the auditor must first use our MDP model to construct all the possible behaviors that the privacy policy allows and then compare it with all the behaviors of the auditee that could have resulted in the observed auditing log. Section 5 presents an algorithm for auditing based on our formal definitions, illustrating the relevance of our work.

The semantics discussed thus far is sufficient to put the previous work on enforcing privacy policies on firm semantic ground. In Section 6, we do so and discuss the strengths and weaknesses of each such approach. In particular, we find that each approach may be viewed as a method of enforcing the policy given the set of all possible allowed behaviors, an intermediate result of our analysis. We compare the previous auditing approaches, which differ in their trade-offs between auditing complexity and accuracy of representing this set of behaviors.

Most auditees are actually interested in multiple purposes and select plans that simultaneously satisfy as many of the desired purposes as possible. Handling the interactions between purposes complicates our semantics. In particular, actions selected by a single plan may be for different purposes. In Section 7, we present examples showing when our semantics can extend to handle multiple purposes and when difficulties arise in determining which purposes an action is for when an auditee is attempting to satisfy various purposes at once. Currently, the state-of-the-art in the understanding of human planning limits our abilities to improve upon our semantics. However, as this understanding improves, one may replace our MDP-like formalism with more detailed ones while retaining our general framework of defining purpose in terms of planning.

We end by discussing other related work, future work, and conclusions. Our contributions include:

- The first semantic formalism of when a sequence of actions is for a purpose,
- An auditing algorithm for this formalism,
- The resituating of previous policy enforcement methods in our formalism and a comparative study of their expressiveness, and

- The first attempt to formally consider the effects on auditing caused by interactions among multiple purposes.

Although motivated by our goal to formalize the notions of *use* and *purpose* prevalently found in privacy policies, our work is more generally applicable to a broad range of policies, such as fiscal policies governing travel reimbursement.

2 Motivation of Our Approach

We start with an informal example that suggests that *an action is for a purpose if the action is part of a plan for achieving that purpose*. Consider a physician working at a hospital who, as a specialist, also owns a private practice that tests for bone damage using a novel technique for extracting information from X-ray images. After seeing a patient and taking an X-ray, the physician forwards the patient's medical record including the X-ray to his private practice to apply this new technology. As this action entails the transmission of protected health information, the physician will have violated HIPAA if this transmission is not for one of the purposes HIPAA allows. The physician would also run afoul of the hospital's own policies governing when outside consultations are permissible unless this action was for a legitimate purpose. Finally, the patient's insurance will only reimburse the costs associated with this consultation if a medical reason (purpose) exists for them. The physician claims that this consultation was for reaching a diagnosis. As such, it is for the purpose of treatment and, therefore, allowed under each of these policies. The hospital auditor, however, has selected this action for investigation since the physician's making a referral to his own private practice makes the alternate motivation of profit possible.

Whether or not the physician violated these policies depends upon details not presented in the above description. For example, we would expect the auditor to ask questions such as: (1) Was the test relevant to the patient's condition? (2) Did the patient benefit medically from having the test? (3) Was this the best option for the patient? We will introduce these details as we introduce each of the factors relevant to the purposes behind the physician's actions.

States and Actions. Sometimes the purposes for which an action is taken depend upon the previous actions and the state of the system. In the above example, whether or not the test is relevant depends upon the condition of the patient, that is, the state that the patient is in.

While an auditor could model the act of transmitting the record as two (or more) different actions based upon the state of the patient, modeling two concepts with one formalism could introduce errors. A better approach is to model the state of the system. The state captures the context in which the physician takes an action and allows for the purposes of an action to depend upon the actions that precede it.

The physician's own actions also affect the state of the system and, thus, the purposes for which his actions are. For example, had the physician transmitted the patient's medical record before taking the X-ray, then the transmission could not have been for treatment since the physician's private practice only operates on X-rays and would have no use for the record without the X-ray.

The above example illustrates that when an action is for a purpose, the action is part of a sequence of actions that can lead to a state in which some goal associated with the purpose is achieved. In the example, the goal is reaching a diagnosis. Only when the X-ray is first added to the record is this goal reached.

Non-redundancy. Some actions, however, may be part of such a sequence without actually being for the purpose. For example, suppose that the patient's X-ray clearly shows the patient's problem. Then, the physician can reach a diagnosis without sending the record to the private practice. Thus, while both taking the X-ray and sending the medical record might be part of a sequence of actions that leads to achieving a diagnosis, the transmission does not actually contribute to achieving the diagnosis: the physician could omit it and the diagnosis could still be reached.

From this example, it may be tempting to conclude that an action is *for* a purpose only if that action is *necessary* to achieve that purpose. However, consider a physician who has a choice between two specialists to whom to send the medical record and must do so to reach a diagnosis. In this scenario, the physician's sending the record to the first specialist is not necessary since he could send it to the second. Likewise, sending the record to the second specialist is not necessary. Yet, the physician must send the record to one or the other specialist and that transmission will be for the purpose of diagnosis. Thus, an action may be for a purpose without being necessary for achieving the purpose.

Rather than *necessity*, we use the weaker notion of *non-redundancy* found in work on the semantics of *causation* (e.g., [Mac74]). Given a sequence of actions that achieves a goal, an action in it is *redundant* if that sequence with that action removed (and otherwise unchanged) also achieves the goal. An action is *non-redundant* if removing that action from the sequence would result in the goal no longer being achieved. Thus, non-redundancy may be viewed as necessity under an otherwise fixed sequence of actions.

For example, suppose the physician decides to send the medical record to the first specialist. Then, the sequence of actions modified by removing this action would not lead to a state in which a diagnosis is reached. Thus, the transmission of the medical record to the first specialist is non-redundant. However, had the X-ray revealed to the physician the diagnosis without needing to send it to a specialist, the sequence of actions that results from removing the transmission from the original sequence would still result in a diagnosis. Thus, the transmission would be redundant.

Quantitative Purposes. Above we implicitly presumed that the diagnosis from each specialist had equal quality. This need not be the case. Indeed, many purposes are actually fulfilled to varying degrees. For example, the purpose of marketing is never completely achieved since there is always more marketing to do. Thus, we model a purpose by assigning to each state-action pair a number that describes how well that action fulfills that purpose when performed in that state. We require that the physician selects the test that maximizes the quality of the diagnosis as determined by total purpose score accumulated over all his actions.

Probabilistic Systems. The success of many medical tests and procedures is probabilistic. For example, with some probability the physician's test may fail to reach a diagnosis. The physician would still have transmitted the medical record for the purpose of diagnosis even if the test failed to reach one. This possibility affects our semantics of purpose: now an action may be for a purpose even if that purpose is never achieved.

To account for such probabilistic events, we model the environment in which the physician operates as probabilistic. For an action to be for a purpose, we require that there be a non-zero probability of the purpose being achieved and that the physician attempts to maximize the expected reward. In essence, we require that the physician attempts to achieve a diagnosis. Thus, the auditee's *plan* determines the purposes behind his actions rather than just the actions themselves.

3 Planning for a Purpose

In this section, we present a formalism for planning that accounts for quantitative purposes, probabilistic systems and non-redundancy. We start by modeling the environment in which the auditee operates as a Markov Decision Process (MDP)—a natural model for probabilistic systems. The reward function of the MDP quantifies the degree of satisfaction of a purpose upon taking an action from a state. If the auditee is motivated to action by only that purpose, then the auditee's actions must correspond to an optimal *plan* for this MDP and these actions are *for* that purpose. We develop a stricter definition of optimal than standard MDPs, which we call NMDPs for *Non-redundant MDP*, to reject redundant actions that neither decrease nor increase the total reward. We end with an example illustrating the use of an NMDP to model an audited environment.

3.1 Markov Decision Processes

An MDP may be thought of as a probabilistic automaton where transitions are labeled with a reward in addition to an action. Rather than having accepting or goal states, the "goal" of a MDP is maximizing the total reward over time.

An MDP is a tuple $m = \langle \mathcal{Q}, \mathcal{A}, t, r, \gamma \rangle$ where $\mathcal{Q}$ is a set of states, $\mathcal{A}$ is a set of actions, $t : \mathcal{Q} \times \mathcal{A} \to \mathcal{D}(\mathcal{Q})$ a transition function from a state and an action to a distribution over states (represented as $\mathcal{D}(\mathcal{Q})$), $r : \mathcal{Q} \times \mathcal{A} \to \mathbb{R}$ a reward function, and γ a discount factor such that $0 < \gamma < 1$. For each state q in $\mathcal{Q}$, the agent using the MDP to plan selects an action a from $\mathcal{A}$ to perform. Upon performing the action a in the state q, the agent receives the reward $r(q, a)$. The environment then transitions to a new state q' with probability $\mu(q')$ where μ is the distribution provided by $t(q, a)$. The goal of the agent is to select actions to maximize its expected total discounted reward $\mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i \rho_i\right]$ where $i \in \mathbb{N}$ (the set of natural numbers) ranges over time modeled as discrete steps, ρ_i is the reward at time i, and the expectation is taken over the probabilistic transitions.

We formalize the agent's plan as a *stationary strategy* (commonly called a "policy", but we reserve that word for privacy policies). A stationary strategy is a function σ from the state space $\mathcal{Q}$ to the set $\mathcal{A}$ of actions (i.e., $\sigma : \mathcal{Q} \to \mathcal{A}$) such that at a state q in $\mathcal{Q}$, the agent always selects to perform the action $\sigma(q)$. Given a strategy σ for an MDP m, its expected total discounted reward is

$$V_m(\sigma, q) = r(q, \sigma(q)) + \gamma \sum_{q' \in \mathcal{Q}} t(q, \sigma(q))(q') * V_m(\sigma, q')$$

The agent selects one of the strategies that optimizes this equation. We denote this set of optimal strategies as $\mathsf{opt}(\langle \mathcal{Q}, \mathcal{A}, t, r, \gamma \rangle)$, or when the transition system is clear from context, as $\mathsf{opt}(r)$. Such strategies are sufficient to maximize the agent's expected total discounted reward despite only depending upon the MDP's current state.

Given the strategy σ and the actual results of the probabilistic transitions yielded by t, the agent exhibits an *execution*. We represent this execution as an infinite sequence $e = [q_1, a_1, q_2, a_2, \ldots]$ of alternating states and actions starting with a state, where q_i is the ith state that the agent was in and a_i is the ith action the agent took, for all i in $\mathbb{N}$. We say an execution e is *consistent* with a strategy σ iff $a_i = \sigma(q_i)$ for all i in $\mathbb{N}$ where a_i is the ith action in e and q_i is the ith state in e. We call a finite prefix of an execution a *behavior*. A behavior is consistent with a strategy if it can be extended to an execution consistent with that strategy.

Under this formalism, the auditee plays the role of the agent optimizing the MDP to plan. We presume that each purpose may be modeled as a reward function. That is, we assume the degree

to which a purpose is satisfied may be captured by a function from states and actions to a real number. The higher the number, the higher the degree to which that purpose is satisfied. When the auditee wants to plan for a purpose p, it uses a reward function, r^p, such that $r^p(q, a)$ is the degree to which taking the action a from state q aids the purpose p. We also assume that the expected total discounted reward can capture the degree to which a purpose is satisfied over time. We say that the auditee plans *for* the purpose p when the auditee adopts a strategy σ that is optimal for the MDP $\langle \mathcal{Q}, \mathcal{A}, t, r^p, \gamma \rangle$. The appendix provides additional background information on MDPs.

3.2 Non-redundancy

MDPs do not require that strategies be non-redundant. Even given that the auditee had an execution e from using a strategy σ in $\mathsf{opt}(r^p)$, some actions in e might not be *for* the purpose p. The reason is that some actions may be redundant despite being costless. The MDP optimization criterion behind opt prevents redundant actions from delaying the achievement of a goal as the reward associated with that goal would be further discounted making such redundant actions sub-optimal. However, the optimization criterion is not affected by redundant actions when they appear after all actions that provide non-zero rewards. Intuitively, the hypothetical agent planning only for the purpose in question would not perform such unneeded actions even if they have zero reward. Thus, to create our formalism of non-redundant MDPs (NMDPs), we replace opt with a new optimization criterion opt^* that prevents these redundant actions while maintaining the same transition structure as a standard MDP.

To account for redundant actions, we must first contrast that with doing nothing. Thus, we introduce a distinguished action N that stands for doing nothing. For all states q, N labels a transition with zero reward (i.e., $r(q, \mathsf{N}) = 0$) that is a self-loop (i.e., $t(q, \mathsf{N})(q) = 1$). (We could put N on only the subset of states that represent possible stopping points by slightly complicating our formalism.) Since we only allow deterministic stationary strategies and N only labels self-loops, this decision is irrevocable: once nothing is done, it is done forever. As selecting to do nothing results in only zero rewards henceforth, it may be viewed as stopping with the previously acquired total discounted reward.

Given an execution e, let $\mathsf{active}(e)$ denote the prefix of e before the first instance of the nothing actions. $\mathsf{active}(e)$ will be equal to e in the case where e does not contain the nothing action.

We use the idea of *nothing* to make formal when one execution intuitively contain more actions than another despite both being of infinite length. An execution e_1 is a *proper sub-execution* of an execution e_2 if and only if $\mathsf{active}(e_1)$ is a proper subsequence of $\mathsf{active}(e_2)$ using the standard notion of subsequence. Note if e_1 does not contain the nothing action, it cannot be a proper sub-execution of any execution.

To compare strategies, we construct all the executions they could produce. To do so, let a *contingency* κ be a function from $\mathcal{Q} \times \mathcal{A} \times \mathbb{N}$ to $\mathcal{Q}$ such that $\kappa(q, a, i)$ is the state that results from taking the action a in the state q the ith time. We say that a contingency κ is *consistent* with an MDP iff κ only picks states to which the transition function t of the MDP assigns a non-zero probability to (i.e., for all q in $\mathcal{Q}$, a in $\mathcal{A}$, and i in $\mathbb{N}$, $t(q, a)(\kappa(q, a, i)) > 0$). Given an MDP m, let $m(q, \kappa)$ be the possibly infinite state model that results of having κ resolve all the probabilistic choices in m and having the model start in state q. Let $m(q, \kappa, \sigma)$ denote the execution that results from using the strategy σ and state q in the non-probabilistic model $m(q, \kappa)$. Henceforth, we only consider contingencies consistent with the model under discussion.

Given two strategies σ and σ', we write $\sigma' \prec \sigma$ if and only if for all contingencies κ and states

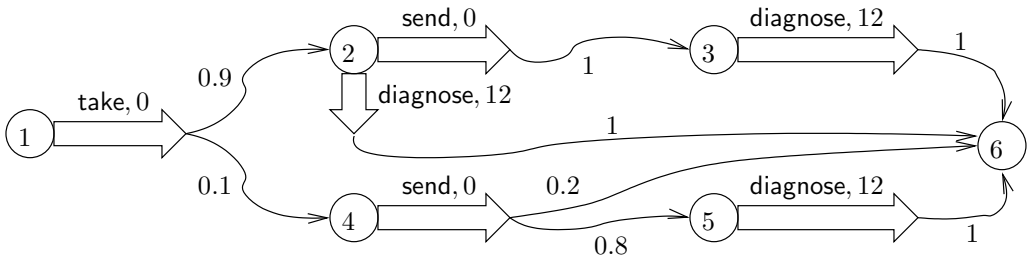

Figure 1: The environment model m_{ex} that the physician used. Circles represent states, block arrows denote possible actions, and squiggly arrows denote probabilistic outcomes. Self-loops of zero reward under all actions, including the special action N, are not shown.

q, $m(q, \kappa, \sigma')$ is a proper sub-execution of or equal to $m(q, \kappa, \sigma)$, and for at least one contingency κ' and state q', $m(q', \kappa', \sigma')$ is a proper sub-execution $m(q', \kappa', \sigma)$. Intuitively, σ' proves that σ produces a redundant execution under κ' and q'. We define $\mathsf{opt}^*(r)$ to be the subset of $\mathsf{opt}(r)$ holding only strategies σ such that for no $\sigma' \in \mathsf{opt}(r)$ does $\sigma' \prec \sigma$. The following theorem, proved in the appendix, shows that non-redundant optimal strategies always exist.

Theorem 1. *For all environment models m, $\mathsf{opt}^*(m)$ is not empty.*

3.3 Example

Suppose an auditor is inspecting a hospital and comes across a physician referring a medical record to his own private practice for analysis of an X-ray as described in Section 2. As physicians may only make such referrals for the purpose of treatment (treat), the auditor may find the physician's behavior suspicious. To investigate, the auditor may formally model the hospital using our formalism.

The auditor would construct the NMDP $m_{\mathsf{ex}} = \langle Q_{\mathsf{ex}}, A_{\mathsf{ex}}, t_{\mathsf{ex}}, r_{\mathsf{ex}}^{\mathsf{treat}}, \gamma_{\mathsf{ex}} \rangle$ shown in Figure 1. The figure conveys all components of the NMDP except γ_{ex}. For instance, the block arrow from the state 1 labeled take and the squiggly arrows leaving it denote that after the agent performs the action take from state 1, the environment will transition to the state 2 with probability 0.9 and to state 4 with probability of 0.1 (i.e., $t_{\mathsf{ex}}(1, \mathsf{take})(2) = 0.9$ and $t_{\mathsf{ex}}(1, \mathsf{take})(4) = 0.1$). The number over the block arrow further indicates the degree to which the action satisfies the purpose of treat. In this instance, it shows that $r_{\mathsf{ex}}^{\mathsf{treat}}(1, \mathsf{take}) = 0$. This transition models the physician taking an X-ray. With probability 0.9, he is able to make a diagnosis right away (from state 2); with probability 0.1, he must send the X-ray to his practice to make a diagnosis. Similarly, the transition from state 4 models that his practice's test has a 0.8 success rate of making a diagnosis; with probability 0.2, no diagnosis is ever reached.

Using the model, the auditor computes $\mathsf{opt}(r_{\mathsf{ex}}^{\mathsf{treat}})$, which consists of those strategies that maximizes the expected total discounted degree of satisfaction of the purpose of treatment where the expectation is over the probabilistic transitions of the model. $\mathsf{opt}(r_{\mathsf{ex}}^{\mathsf{treat}})$ includes the appropriate strategy σ_1 where $\sigma_1(1) = \mathsf{take}$, $\sigma_1(4) = \mathsf{send}$, $\sigma_1(2) = \sigma_1(3) = \sigma_1(5) = \mathsf{diagnose}$, and $\sigma_1(6) = \mathsf{N}$. Furthermore, $\mathsf{opt}(r_{\mathsf{ex}}^{\mathsf{treat}})$ excludes the redundant strategy σ_2 that performs a redundant send where σ_2 is the same as σ_1 except for $\sigma_2(2) = \mathsf{send}$. Performing the extra action send delays the reward

of 12 for achieving a diagnosis resulting in its discounted reward being $\gamma_{\mathsf{ex}}^2 * 12$ instead of $\gamma_{\mathsf{ex}} * 12$ and, thus, the strategy is not optimal.

However, $\mathsf{opt}(r_{\mathsf{ex}}^{\mathsf{treat}})$ does include the redundant strategy σ_3 that is the same as σ_1 except for $\sigma_3(6) = \mathsf{send}$. $\mathsf{opt}(r_{\mathsf{ex}}^{\mathsf{treat}})$ includes this strategy despite the send actions from state 6 being redundant since no positive rewards follow the send actions. Fortunately, $\mathsf{opt}^*(r_{\mathsf{ex}}^{\mathsf{treat}})$ does not include σ_3 since σ_1 is both in $\mathsf{opt}(r_{\mathsf{ex}}^{\mathsf{treat}})$ and $\sigma_1 \prec \sigma_3$. To see that $\sigma_1 \prec \sigma_3$ note that for every contingency κ and state q, the $m_{\mathsf{ex}}(q, \kappa, \sigma_1)$ has the form b followed by an finite sequence of nothing actions (interleaved with the state 6) for some finite prefix b. For the same κ, $m_{\mathsf{ex}}(q, \kappa, \sigma_3)$ has the form b followed by an infinite sequence of send actions (interleaved with the state 6) for the same b. Thus, $m_{\mathsf{ex}}(q, \kappa, \sigma_1)$ is a proper sub-execution of $m_{\mathsf{ex}}(q, \kappa, \sigma_3)$.

4 Auditing

In the above example, the auditor constructed a model of the environment in which the auditee operates. The auditor must use the model to determine if the auditee obeyed the policy. We first discuss this process for auditing restrictive policy rules and revisit the above example. Then, we discuss the process for prohibitive policy rules. In the next section, we provide an auditing algorithm that automates comparing the auditee's behavior, as recorded in a log, to the set of allowed behaviors.

4.1 Auditing Restrictive Rules

Suppose that an auditor would like to determine whether an auditee performed some logged actions *only for* the purpose p. The auditor can compare the logged behavior to the behavior that a hypothetical agent would perform when planning for the purpose p. In particular, the hypothetical agent selects a strategy from $\mathsf{opt}^*(\langle \mathcal{Q}, \mathcal{A}, t, r^p, \gamma \rangle)$ where $\mathcal{Q}$, $\mathcal{A}$, and t models the environment of the auditee; r^p is a reward function modeling the degree to which the purpose p is satisfied; and γ is an appropriately selected discounting factor. If the logged behavior of the auditee would never have been performed by the hypothetical agent, then the auditor knows that the auditee violated the policy.

In particular, the auditor must consider all the possible behaviors the hypothetical agent could have performed. For a model m, let $\mathsf{behv}^*(r^p)$ represent this set where a finite prefix b of an execution is in $\mathsf{behv}^*(r^p)$ if and only if there exists a strategy σ in $\mathsf{opt}^*(r^p)$, a contingency κ, and a state q such that b is a subsequence of $m(q, \kappa, \sigma)$.

The auditor must compare $\mathsf{behv}^*(r^p)$ to the set of all behaviors that could have caused the auditor to observe the log that he did. We presume that the log ℓ was created by a process log that records features of the current behavior. That is, $\mathsf{log}\colon B \to L$ where B is the set of behaviors and L the set of logs, and $\ell = \mathsf{log}(b)$ where b is the prefix of the actual execution of the environment available at the time of auditing. The auditor must consider all the behaviors in $\mathsf{log}^{-1}(\ell)$ as possible where log^{-1} is the inverse of the logging function. In the best case for the auditor, the log records the whole prefix b of the execution that transpired until the time of auditing, in which case $\mathsf{log}^{-1}(\ell) = \{\ell\}$.

If $\mathsf{log}^{-1}(\ell) \cap \mathsf{behv}^*(r^p)$ is empty, then the auditor may conclude that the auditee did not plan for the purpose p, and, thus, violated the rule that auditee must only perform the actions recorded in ℓ for the purpose p; otherwise, the auditor must consider it possible that the auditee planned for

the purpose p.

If $\mathsf{log}^{-1}(\ell) \subseteq \mathsf{behv}^*(r^p)$, the auditor might be tempted to conclude that the auditee surely obeyed the policy rule. However, as illustrated in the second example below, this is not necessarily true. The problem is that $\mathsf{log}^{-1}(\ell)$ might have a non-empty intersection with $\mathsf{behv}^*(r^{p'})$ for some other purpose p'. In this case, the auditee might have been actually planning for the purpose p' instead of p. Indeed, given the likelihood of such other purposes for non-trivial scenarios, we consider proving compliance practically impossible. However, this incapability is of little consequence: $\mathsf{log}^{-1}(\ell) \subseteq \mathsf{behv}^*(r^p)$ does imply that the auditee is behaving as though he is obeying the policy. That is, in the worse case, the auditee is still doing the right things even if for the wrong reasons.

4.2 Example

Below we revisit the example of Section 3.3. We consider two cases. In the first, the auditor shows that the physician violated the policy. In the second, auditing is inconclusive.

Violation Found. Suppose after constructing the model as above in Section 3.3, the auditor maps the actions recorded in the access log ℓ_1 to the actions of the model m_{ex}, and finds $\mathsf{log}^{-1}(\ell_1)$ holds only a single behavior: $b_1 = [1, \mathsf{take}, 2, \mathsf{send}, 3, \mathsf{diagnose}, 6, \mathsf{N}, 6]$. Next, using $\mathsf{opt}^*(r_{\mathsf{ex}}^{\mathsf{treat}})$, as computed above, the auditor constructs the set $\mathsf{behv}^*(r_{\mathsf{ex}}^{\mathsf{treat}})$ of all behaviors an agent planning for treatment might exhibit. The auditor would find that b_1 is not in $\mathsf{behv}^*(r_{\mathsf{ex}}^{\mathsf{treat}})$.

To see this, note that every execution e_1 that has b_1 as a prefix is generated from a strategy σ such that $\sigma(2) = \mathsf{send}$. The strategy σ_2 from Section 3.3 is one such strategy. None of these strategies are members of $\mathsf{opt}(r_{\mathsf{ex}}^{\mathsf{treat}})$ for the same reason as σ_2 is not a member. Thus, b_1 cannot be in $\mathsf{behv}^*(r_{\mathsf{ex}}^{\mathsf{treat}})$. As $\mathsf{log}^{-1}(\ell) \cap \mathsf{behv}^*(r_{\mathsf{ex}}^{\mathsf{treat}})$ is empty, the audit reveals that the physician violated the policy.

Inconclusive. Now suppose that the auditor sees a different log ℓ_2 such that $\mathsf{log}^{-1}(\ell_2) = \{b_2\}$ where $b_2 = [1, \mathsf{take}, 4, \mathsf{send}, 5, \mathsf{diagnose}, 6, \mathsf{N}, 6]$. In this case, our formalism would not find a violation since b_2 is in $\mathsf{behv}^*(r_{\mathsf{ex}}^{\mathsf{treat}})$. In particular, the strategy σ_1 from above produces the behavior b_2 under the contingency that selects the bottom probabilistic transition from state 1 to state 4 under the action take.

Nevertheless, the auditor cannot be sure that the physician obeyed the policy. For example, consider the NMDP m'_{ex} that is m_{ex} altered to use the reward function $r_{\mathsf{ex}}^{\mathsf{profit}}$ instead of $r_{\mathsf{ex}}^{\mathsf{treat}}$. $r_{\mathsf{ex}}^{\mathsf{profit}}$ assigns a reward of zero to all transitions except for the send actions from states 2 and 4, to which it assigns a reward of 9. σ_1 is in $\mathsf{opt}^*(r_{\mathsf{ex}}^{\mathsf{profit}})$ meaning that not only the same actions (those in b_2), but even the exact same strategy can be either for the allowed purpose treat or the disallowed purpose profit. Thus, if the physician did refer the record to his practice for profit, he cannot be caught as he has tenable deniability of his ulterior motive of profit.

4.3 Auditing Prohibitive Rules

In the above example, the auditor was enforcing the rule that the physician's actions be *only for* treatment. Now, consider auditing to enforce the rule the that physician's actions are *not for* personal profit. After seeing the log ℓ, the auditor could check whether $\mathsf{log}^{-1}(\ell) \cap \mathsf{behv}^*(r_{\mathsf{ex}}^{\mathsf{profit}})$ is empty. If so, then the auditor knows that the policy was obeyed. If not, then the auditor cannot prove nor disprove a violation. In the above example, just as the auditor is unsure whether the

actions were *for* the required purpose of treatment, the auditor is unsure whether the actions are *not for* the prohibited purpose of profit.

An auditor might decide to investigate some of the cases where $\mathsf{log}^{-1}(\ell) \cap \mathsf{behv}^*(r_{\mathsf{ex}}^{\mathsf{profit}})$ is not empty. In this case, the auditor could limit his attention to only those possible violations of a prohibitive rule that cannot be explained away by some allowed purpose. For example, in the inconclusive example above, the physician's actions can be explained with the allowed purpose of treatment. As the physician has tenable deniability, it is unlikely that investigating his actions would be a productive use of the auditor's time. Thus, the auditor should limit his attention to those logs ℓ such that both $\mathsf{log}^{-1}(\ell) \cap \mathsf{behv}^*(r_{\mathsf{ex}}^{\mathsf{profit}})$ is non-empty and $\mathsf{log}^{-1}(\ell) \cap \mathsf{behv}^*(r_{\mathsf{ex}}^{\mathsf{treat}})$ is empty.

A similar additional check using disallowed purposes could be applied to enforcing restrictive rules. However, for restrictive rules, this check would identify cases where the auditee's behavior could have been either for the allowed purpose or a disallowed purpose. Thus, it would serve to find additional cases to investigate and increase the auditor's workload rather than reduce it. Furthermore, the auditee would have tenable deniability for these possible ulterior motives, making these investigations a poor use of the auditor's time.

5 Auditing Algorithm

We would like to automate the auditing process described above. To this end, we present in Figure 2 an algorithm Audit that aids the auditor in comparing the log to the set of allowed behaviors. As we are not interested in the details of the logging process and would like to focus on the planning aspects of our semantics, we limit our attention to the case where $\mathsf{log}(b) = b$. As proved below (Theorem 2), $\textsc{Audit}(m, b)$ returns true if and only if $\mathsf{log}^{-1}(b) \cap \mathsf{behv}^*(m)$ is empty. In the case of a restrictive rule, the auditor may conclude that the policy was violated when Audit returns true. In case of a prohibitive rule, the auditor may conclude the policy was obeyed when Audit returns true.

Audit operates in two steps. The first checks to make sure that the behavior b is not inherently redundant (lines 01–05). If it is, then $\mathsf{log}^{-1}(b) \cap \mathsf{behv}^*(m)$ will be empty and the algorithm returns true. Audit checks b by comparing the actions taken in each state to doing nothing. If the expected total discounted reward for doing nothing in a state q is higher than that for doing the action a in q, then a introduces redundancy into any strategy σ such that $\sigma(q) = a$. Thus, if $b = [\ldots, q, a, \ldots]$, we may conclude that $\mathsf{log}^{-1}(b) \cap \mathsf{behv}^*(m)$ is empty.

The second step compares the optimal values of two MDPs. One of the them is the NMDP m treated as an MDP, which is already optimized during the first step. The other m' is constructed from m (lines 07–17) so that only the actions in the log b are selected during optimization. If the expected total discounted reward of each of these MDPs is unequal, then $\mathsf{log}^{-1}(b) \cap \mathsf{behv}^*(m)$ is empty.

Below we formalize these ideas. Lemma 1 justifies our two step approach while Lemmas 2 and 3 justify how we perform the first and second step, respectively. They allow us to conclude the correctness of our algorithm in Theorem 2. We defer proofs and additional propositions to the appendix.

```
AUDIT(⟨Q, A, t, r, γ⟩, [q_0, a_1, q_1, ... a_n, q_n]):
 01  V*_m := solveMDP(⟨Q, A, t, r, γ⟩)
 02  for(i := 0; i < n; i++):
 03      if(a_{i+1} ≠ N):
 04          if(r[q_i][a_{i+1}] + γ Σ_{j=0}^{|Q|} t[q_i][a_{i+1}][j] * V*_m[j] ≤ 0):
 05              return true
 06  r* := 0
 07  for(j := 0; j < |Q|; j++):
 08      for(k := 0; k < |A|; k++):
 09          r'[j][k] := r[j][k]
 10          if(r* < absoluteValue(r[j][k])):
 11              r* := absoluteValue(r[j][k])
 12  ω := 2 * r*/(1 − γ) + 1
 13  for(i := 0; i < n; i++):
 14      for(k := 0; k < |A|; k++):
 15          if(k ≠ a_{i+1}):
 16              r'[q_i][k] := −ω
 17  m' := ⟨Q, A, t, r', γ⟩
 18  V*_{m'} := solveMDP(⟨Q, A, t, r', γ⟩)
 19  for(j := 0; j < |Q|; j++):
 20      if(V*_m[j] = V*_{m'}[j]):
 21          return false
 22  return true
```

Figure 2: The algorithm AUDIT. solveMDP may be any MDP solving algorithm. The algorithm assumes functions are represented as arrays and states and actions are represented as indexes into these arrays.

5.1 Useless States and the Two Steps

We say an action is *useless* at a state if taking it would always lead to redundancy. Formally, let the set U_m be the subset of $\mathcal{Q} \times \mathcal{A}$ such that $\langle q, a \rangle$ is in U_m if and only if $a \neq \mathsf{N}$ and for all strategies σ, $Q_m(\sigma, q, a) \leq 0$ where $Q_m(\sigma, q, a) = r(q, a) + \gamma \sum_{q'} t(q, a)(q') * V_m(\sigma, q')$.

We call $\langle q, a \rangle$ in set U_m *useless* since any strategy σ such that $\sigma(q) = a$ could be replaced by a strategy σ' that is the same as σ except for having $\sigma'(q) = \mathsf{N}$ without lowering the expected total discounted reward. To make this formal, let $U(\sigma)$ be a strategy such that $U(\sigma)(q) = \mathsf{N}$ if $\langle q, \sigma(q) \rangle \in U$ and $U(\sigma)(q) = \sigma(q)$ otherwise. The following justifies calling these pairs *useless*: for all σ and q, $V_m(\sigma, q) \leq V_m(U_m(\sigma), q)$ (Proposition 1).

We are also interested in the set $\mathsf{strg}(b)$ of strategies that could have resulted in the behavior b: $\mathsf{strg}(b) = \{ \sigma \in \mathcal{Q} \to \mathcal{A} \mid \forall i < n . a_{i+1} = \sigma(q_i) \}$ where $b = [q_0, a_1, q_1, a_2, \ldots, a_n, q_n]$.

Lemma 1. *For all environment models m and all behaviors $b = [q_0, a_1, q_1, \ldots, a_n, q_n]$, $\mathsf{log}^{-1}(b) \cap \mathsf{behv}^*(m)$ is empty if and only if (1) there exists i such that $0 \leq i < n$ and $\langle q_i, a_{i+1} \rangle \in U_m$ or (2) $\mathsf{strg}(b) \cap \mathsf{opt}(m)$ is empty,*

Thus, checking whether $\mathsf{log}^{-1}(b) \cap \mathsf{behv}^*(m)$ is empty has been reduced to checking the two conditions (1) and (2). We explain how to check each of these in the next two sections.

5.2 Step 1: Inherent Redundancy

Rather than construct U_m explicitly, we use the following lemma to check condition (1). The lemma uses the definition $Q^*_m(q, a) = r(q, a) + \gamma \sum_{q'} t(q, a)(q') * V^*_m(q')$ where $V^*_m(q) = \max_\sigma V_m(\sigma, a)$.

Lemma 2. *For all environment models m, states q, and actions a, $\langle q, a \rangle$ is in U_m if and only if $a \neq \mathsf{N}$ and $Q^*_m(q, a) \leq 0$.*

5.3 Step 2: Checking Optimality

To check (2), we construct a model m' from m that limits the optimization to selecting a strategy that can cause the observed behavior b. To do so, we adjust the reward function of m so that the actions taken in b are always taken by the optimal strategies of m'. That is, if $b = [q_0, a_1, q_1, \ldots, a_n, q_n]$, then for each q_i and a_{i+1}, we replace the reward for taking an action a' other than a_{i+1} from the state q with a negative reward $-\omega$ that is so low as to assure that the action a' would not be used by any optimal strategy. We use $\omega > 2r^*/(1-\gamma)$ where r^* is the reward with the largest magnitude appearing in m since the total discounted reward is bounded from below by $-r^*/(1-\gamma)$ and from above by $r^*/(1-\gamma)$ (recall that $\sum_{i=0}^{\infty} \gamma^i r^* = r^*/(1-\gamma)$).

We formally define m' to be $\mathsf{fix}(m, b)$ where $\mathsf{fix}(m, []) = m$ and

$$\mathsf{fix}(\langle \mathcal{Q}, \mathcal{A}, t, r, \gamma \rangle, [q_0, a_1, q_1, \ldots, a_n, q_n]) = \mathsf{fix}(\langle \mathcal{Q}, \mathcal{A}, t, r', \gamma \rangle, [q_1, \ldots, a_n, q_n])$$

where $r'(q_0, a) = -\omega$ for all $a \neq a_1$ and $r'(q_0, a_1) = r(q_0, a_1)$. The construction fix has the following useful property: $\mathsf{strg}(b) \cap \mathsf{opt}(m)$ is empty if and only if $\mathsf{opt}(\mathsf{fix}(m, b)) \cap \mathsf{opt}(m)$ is empty (Proposition 11). This property is useful since testing whether $\mathsf{opt}(m) \cap \mathsf{opt}(\mathsf{fix}(m, b))$ is empty may be reduced to simply comparing their optimal values: $\mathsf{opt}(m) \cap \mathsf{opt}(\mathsf{fix}(m, b))$ is empty if and only if for all states q, $\max_\sigma V_{\mathsf{fix}(m,b)}(\sigma, q) \neq \max_\sigma V_m(\sigma, q)$ (Proposition 12). Fortunately, algorithms exist for finding the optimal value of MDPs (see, e.g., [RN03]).

These two propositions combine to yield the next lemma, which justifies how we conduct testing for the second condition of Lemma 1 in the second step of Audit.

Lemma 3. *For all environment models m and behaviors b,* $\mathsf{strg}(b) \cap \mathsf{opt}(m)$ *is empty if and only if for all q,* $\max_\sigma V_{\mathsf{fix}(m,b)}(\sigma, q) \neq \max_\sigma V_m(\sigma, q)$.

These lemmas combine with reasoning about the actual code of the program to yield its correctness.

Theorem 2. *For all environment models m and behaviors b,* $\textsc{audit}(m, b)$ *returns true if and only if* $\mathsf{log}^{-1}(b) \cap \mathsf{behv}^*(m)$ *is empty.*

The running time of the algorithm is dominated by the two MDP optimizations. These may be done exactly by reducing the optimization to a system of linear equations [d'E63]. Such systems may be solved in polynomial time [Kha79, Kar84]. However, in practice, large systems are often difficult to solve. Fortunately, a large number of algorithms for making iterative approximations exist whose run time depends on the quality of the approximation. (See [LDK95] for a discussion.)

6 Applying our Formalism to Past Methods

Past methods of enforcing purpose requirements have not provided methods of assigning purposes to sequences of actions. Rather, they presume that the auditor (or someone else) already has a method of determining which behaviors are for a purpose. In essence, these methods presuppose that the auditor already has the set of allowed behaviors $\mathsf{behv}^*(r^p)$ for the purpose p that he is enforcing. These methods differ in their intensional representations of the set $\mathsf{behv}^*(r^p)$. Thus, some may represent a given set exactly while others may only be able to approximate it. These differences mainly arise from the different mechanisms they use to ensure that the auditee only exhibits behaviors from $\mathsf{behv}^*(r^p)$. We use our semantics to study how reasonable these approximations are.

Byun et al. use role-based access control [San96] to consider purposes [BBL05, BL08, NBL+10]. They associate purposes with sensitive resources and with roles, and their method only grants the user access to the resource when the purpose of the user's role matches the resource's purpose. The method does not, however, explain how to determine which purposes to associate with which roles. Furthermore, a user in a role can perform actions that do not fit the purposes associated with his role allowing him to use the resource for a purpose other than the intended one. Thus, their method is only capable of enforcing policies when there exists some subset A of the set of actions $\mathcal{A}$ such that $\mathsf{behv}^*(r^p)$ is equal to the set of all interleavings of A with $\mathcal{Q}$ of finite but unbounded length (i.e., $\mathsf{behv}^*(r^p) = (\mathcal{Q} \times A)^*{:}\mathcal{Q}$ where : is append raised to work over sets in the standard pairwise manner). The subset A corresponds to those actions that use a resource with the same purpose as the auditee's role. Despite these limitations, their method can implement the run-time enforcement used at some organizations, such as a hospital that allows physicians access to any record to avoid denying access in time-critical emergencies. However, it does not allow for the fine-grain distinctions used during post-hoc auditing done at some hospitals to ensure that physicians do not abuse their privileges.

Al-Fedaghi uses the work of Byun et al. as a starting point but concludes that rather than associating purposes with roles, one should associate purposes with sequences of actions [AF07]. Influenced by Al-Fedaghi, Jafari et al. adopt a similar position calling these sequences *workflows* [JSNS09]. The set of workflows allowed for a purpose p corresponds to $\mathsf{behv}^*(r^p)$. They do not provide a formal method of determining which workflows belong in the allowed set. They do

not consider probabilistic transitions and the intuition they supply suggests that they would only include workflows that successfully achieves or improves the purpose. Thus, our approach appears more lenient by including some behaviors that fail to improve the purpose.

Others have adopted a hybrid approach allowing for the roles of an auditee to change based on the state of the system [PGY08, EKWB11]. These changes effectively allow role-based access control to simulate the workflow methods to be just as expressive while introducing a level of indirection inhabited by dynamic roles.

Agrawal et al. use a *query intrusion model* to enforce purpose requirements that operates in a manner similar to intrusion detection [AKSX02]. Their method flags a request for access as a possible violation if the request claims to be for a purpose despite being dissimilar to previous requests for the same purpose. To avoid false positives, the set of allowed behaviors $\mathsf{behv}^*(r^p)$ would have to be small or have a pattern that the query intrusion model could recognize.

Jif is a language extension to Java designed to enforce requirements on the flows of information in a program [CMVZ09]. Hayati and Abadi explain how to reduce purpose requirements to information flow properties that Jif can enforce [HA05]. Their method requires that inputs are labeled with the purposes for which the policy allows the program to use them and that each unit of code be labeled with the purposes for which that code operates. If information can flow from an input statement labeled with one purpose to code labeled for a different purpose, their method produces a compile-time type error. (For simplicity, we ignore their use of sub-typing to model sub-purposes.) In essence, their method enforces the rule *if information i flows to code c, then i and c must be labeled with the same purpose.* The interesting case is when the code c uses the information i to perform some observable action $a_{c,i}$, such as producing output. Under our semantics, we treat the program as the auditee and view the policy as limiting these actions. By directly labeling code, their method does not consider the contexts in which these actions occur. Rather the action $a_{c,i}$ is aways either allowed or not based on the purpose labels of c and i. By not considering context, their method is subject to the same limitations as the method of Byun et al. with the subset A being equal to the set of all actions $a_{c,i}$ such that c and i have the same label. However, using more advanced type systems (e.g., typestate [SY86]), they might be able extend their method to consider the context in which code is executed and increase the method's expressiveness.

7 Multiple Purposes

So far, our formalism allows our hypothetical agent to consider only a single purpose. However, auditees may perform an action for more than one purpose. In many cases, the auditor may simply ignore any action that is not governed by the privacy policy and not relevant to the plans the auditee is employing that uses governed actions.

In the physician example above, the physician already implicitly considered many other purposes before even seeing this current patient. For example, the physician presumably performed many actions not mentioned in the model in between taking the X-ray, sending it, and making a diagnosis, such as going on a coffee break. As these actions are not governed by the privacy policy and neither improves nor degrades the diagnosis even indirectly, the auditor may safely ignore them. Thus, our semantics can handle multiple purposes in this limited fashion.

However, in other cases, the interactions between purposes become important. Below we discuss two complementary ways that an auditee can consider multiple purposes that produce interactions. In the first, the auditee considers one purpose after another. In the second, the auditee attempts to

optimize for multiple purposes simultaneously. We find that our semantics may easily be extended to handle the first, but difficulties arise for the second. We end the section by considering what features a formalism would need to handle simultaneous consideration of purposes and the challenges they raise for auditing.

7.1 Sequential Consideration

Yahoo!'s privacy policy states that they will not contact children for the purpose of marketing [Yah10a]. Suppose Yahoo! decides to change the name of `games.yahoo.com` to `fun.yahoo.com` because they believe the new name will be easier to market. They notify users of `games.yahoo.com`, including children, of the upcoming change so that they may update their bookmarks.

In this example, the decision to change names, made for marketing, causes Yahoo! to contact children. However, we do not feel this is a violation of Yahoo!'s privacy policy. A decision made for marketing altered the expected future of Yahoo! in such a way that customer service would suffer if Yahoo! did nothing. Thus, to maintain good customer service, Yahoo! made the decision to notify users without further consideration of marketing. Since Yahoo! did not consider the purpose of marketing while making this decision, contacting the children was not *for* marketing despite Yahoo! considering the implications of changing the name for marketing while making its decision to contact children.

Bratman describes such planning in his work formalizing *intentions* [Bra87]. He views it as a sequence of planning steps in which the intention to act (e.g., to change the name) at one step may affect the plans formed at later steps. In particular, each step of planning starts with a model of the environment that is refined by the intentions formed by each of the previous planning steps. The step then creates a plan for a purpose that further refines the model with new intentions resulting from this plan. Thus, a purpose of a previous step may affect the plan formed in a later step for a different purpose by constraining the choices available at the later step of planning. We adopt the stance that an action selected at a step is *for* the purpose optimized at that step but not other previous purposes affecting the step.

7.2 Simultaneous Consideration

At other times, an auditee might consider more than one purpose in the same step. For example, the physician may have to both provide quality treatment and respect the patient's financial concerns. In this case, the physician may not be able to simultaneously provide the highest quality care at the lowest price. The two competing concerns must be balanced and the result may not maximize the satisfaction of either of them.

The traditional way of modeling the simultaneous optimization of multiple rewards is to combine them into a single reward using a weighted average over the rewards. Each reward would be weighted by how important it is to the auditee performing the optimization. This amalgamation of the various purpose rewards makes it difficult to determine for which purpose various actions are selected.

One possibility is to analyze the situation using counterfactual reasoning (see, e.g., [Mac74]). For example, given that the auditee performed an action a while optimizing a combination of purposes p_1 and p_2, the auditor could ask if the auditee would have still performed the action a even if the auditee had not considered the purpose p_1 and had only optimized the purpose p_2. If

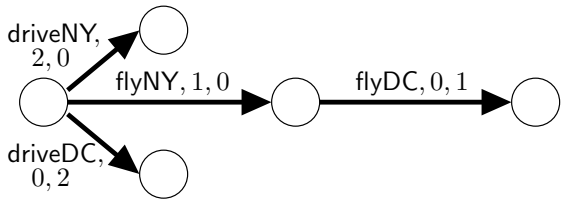

Figure 3: Model of a traveler deciding whether to fly or drive. Since every transition is deterministic, we represent each as a single arrow. Each is labeled with the action name, the rewards for business and the rewards for lecturing in that order. Self-loops of zero reward are not shown including all those labeled with the nothing action N.

not, than the auditor could determine that the action was for p_1. However, as the next example shows, such reasoning is not sufficient to determine the purposes of the actions.

To show the generality of purposes, we consider an example involving travel reimbursement. Consider a Philadelphian who needs to go to New York City for a business meeting with his employer and is invited to give a lecture at a conference in Washington, D.C., with his travel expenses reimbursed by the conference. He could drive to either New York or Washington (modeled as the actions driveNY and driveDC, respectively). However, due to time constraints he cannot drive to both of them. To attend both events, he needs to fly to both (modeled as actions flyNY and flyDC). As flying is more expensive, both driving actions receives a higher reward than flying (2 instead of 1), but flying is better than not going (0). Figure 3 models the traveler's environment.

Given these constraints, he decides to fly to both only to find auditors at both events scrutinizing his decision. For example, an auditor working for the conference could find that his flight to Washington was not for the lecture since the traveler would have driven had it not been for work. If the conference's policy requires that reimbursed flights are *only for* the lecture, the auditor might deny reimbursement. However, the employer seems even less likely to reimburse the traveler for his flight to Washington since the flight is redundant for getting to New York.

However, under the semantics discussed above, each flight would be for both purposes since only when the traveler considers both does he decide to take either flight. While having the conference reimburse the traveler for his flight to Washington seems reasonable, the idea that they should also reimburse him for his flight to New York appears counterintuitive.

Our approach of sequential planning also cannot explain this example. To plan sequentially, the traveler must consider one of the two events first. If, for example, he considers New York first, he will decided to drive to New York and then decline the invitation to Washington. Only by considering both events at once, does he decide to fly.

We believe resolving this conflict requires extending our semantics to consider requirements that an action be *for* a purpose (as opposed to *not for* or *only for*). Furthermore, we believe that the optimization of combinations of purposes does not accurately model human planning with multiple purposes. Intuitively, the traveler selects flyDC not *for* work but also not *only for* the conference. Rather flyDC seems be *for* the conference under the constraint that it must not prevent the traveler from attending the meeting. In the next section, we consider the possibility of modeling human planning more accurately.

7.3 Modeling Human Planning

While MDPs are useful for automated planning, they are not specialized for modeling planning by humans, leading to the search for more tailored models [Sim55, GS02]. Simon proposed to model humans as having *bounded rationality* to account for their limitations and their lack of information [Sim55]. Work on formalizing bounded rationality has resulted in a variety of planning principles ranging from the systematic (e.g., Simon's *satisficing*) to the heuristic (e.g., [Gig02]). However, "[a] comprehensive, coherent theory of bounded rationality is not available" [Sel02, p14] and there still is "a significant amount of unpredictability in how an animal or a human being will undertake to solve a problem" such as planning [DKP96, p40].

We view creating semantics more closely tied to human planning interesting future work. However, modeling human planning may prove complex enough to justify accepting the imperfections of semantics such as ours or even heuristic based approaches for finding violations such as the query intrusion model discussed above [AKSX02].

Despite these difficulties, one could look for discrepancies between a semantics of purpose requirements and experimental results on planning. In this manner one could judge how closely a semantics approximates human planning in the ways relative to purpose requirements.

In particular, our semantics appears to hold human auditees to too high of a standard: they are unlikely to always be able to pick the optimal strategy for a purpose. When enforcing a restrictive rule, this strictness could result in the auditor investigating some auditees who honestly planned for the only allowed purpose, but failed to find the optimal policy. While such investigations would be false positives, they do have the pleasing side-effect of highlighting areas in which an auditee could improve his planning.

In the case of enforcing prohibitive rules, this strictness could cause the auditor to miss some violations that do not optimize the prohibited purpose, but, nevertheless, are for the purpose. The additional checks proposed at the end of Section 4.3 could be useful for detecting these violations: if the auditee's actions are not consistent with a strategy that optimizes any of the allowed purposes but does improve to some degree the prohibited purpose, the actions may warrant extra scrutiny.

While our semantics is limited by our understanding of human planning, it still reveals concepts crucial to the meaning of *purpose*. Ideas such as planning and non-redundancy will guide future investigations on the topic.

8 Related Work

We have already covered the most closely related work in Section 6. Below we discuss work on related problems and work on purpose from other fields.

Minimal Disclosure. The works most similar to ours in approach have been on *minimal disclosure*, which requires that the amount of information used in granting a request for access should be as little as possible while still achieving the purpose behind the request. Massacci, Mylopoulos, and Zannone define minimal disclosure for Hippocratic databases [MMZ06]. Barth, Mitchell, Datta, and Sundaram study minimal disclosure in the context of workflows [BMDS07]. They model a workflow as meeting a utility goal if it satisfies a temporal logic formula. Minimizing the amount of information disclosed is similar to an agent maximizing his reward and thereby not performing actions that have costs but no benefits. However, in addition to having different research goals, we

consider several factors that these works do not, including quantitative purposes that are satisfied to varying degrees and probabilistic behavior resulting in actions being for a purpose despite the purpose not being achieved.

Expressing Privacy Policies with Purpose. Work on understanding the components of privacy policies has shown that *purpose* is a common component of privacy rules (e.g., [BA05, BA08]). Some languages for specifying access-control policies allow the purpose of an action to partially determine if access is granted [PS03, Cra02, BKKF05, BKK06]. However, these languages do not give a formal semantics to the purposes. Instead they rely upon the system using the policy to determine whether an action is for a purpose or not.

Philosophical Foundations. Taylor provides a detailed explanation of the importance of planning to the meaning of *purpose*, but does not provide any formalism [Tay66].

The sense in which the word "purpose" is used in privacy policies is also related to the ideas of *desire*, *motivation*, and *intention* discussed in works of philosophy (e.g., [Ans57]). The most closely related to our work is that of Bratman's on intentions from which we get our model of sequential planning [Bra87]. In his work, an *intention* is an action an agent plans to take where the plan is formed while attempting to maximize the satisfaction of the agent's *desires*; Bratman's *desires* correspond to our *purposes*. Roy formalized Bratman's work using logics and game theory [Roy08]. However, these works are concerned with when an action is rational rather than determining the purposes behind the action.

We borrow the notion of *non-redundancy* from Mackie's work on formalizing *causality* using counterfactual reasoning [Mac74]. In particular, Mackie defines a *cause* to be a non-redundant part of a sufficient explanation of an effect. Roughly speaking, we replace the causes with actions and the effect with a purpose. The extension to our semantics proposed in Section 7.2, may be seen as another instance of non-redundancy. This time, we replace the causes with purposes and the effect with an action. This suggests that for an action to be for a purpose, we expect both that the action was non-redundant for improving that purpose and that the purpose was non-redundant in motivating the action. That is, we expect planning to be parsimonious.

Planning. Psychological studies have produced models of human thought (e.g., [ABB$^+$04]). However, these are too low-level and incomplete for our needs [DKP96]. The GOMS formalism provides a higher level model, but is limited to selecting behavior using simple planning approaches [JK96]. Simon's approach of *bounded rationality* [Sim55] and related heuristic-based approaches [GS02] model more complex planning, but with less precise predictions.

9 Conclusions and Future Work

We use planning to present the first formal semantics for determining when a sequence of actions is for a purpose. In particular, our semantics uses an MDP-like model for planning, which allows us to automate auditing for both restrictive and prohibitive purpose requirements. Furthermore, our semantics highlights that an action can be for a purpose even if that purpose is never achieved, a point present in philosophical works on the subject (e.g., [Tay66]), but whose ramifications on policy enforcement had been unexplored. Lastly, our framework allows us to explain and compare previous methods of policy enforcement in terms of a formal semantics.

However, we recognize the limitations of this model: it imperfectly models human planning and only captures some forms of planning for multiple purposes. Nevertheless, we believe the essence of our work is correct: an action is for a purpose if the actor selects to perform that action while planning for the purpose. Future work will instantiate our semantic framework with more complete models of human planning.

Fundamentally, our work shows the difficulties of enforcement due to issues such as the tenable deniability of ulterior motives. These difficulties justify policies prohibiting conflicts of interest and requiring the separation of duties despite possibly causing inefficiencies. For example, many hospitals would err on the side of caution and disallow referral from a physician to his own private practice or require a second opinion to do so, thereby restraining the ulterior motive of profit. Indeed, despite the maxim that *privacy is security with a purpose*, due to these difficulties, purpose possibly plays the role of guidance in crafting more operational internal policies that organizations enforce rather than the role of a direct input to the formal auditing process itself. In light of this possibility, one may view our work as a way to judge the quality of these operational policies relevant to the intent of the purpose requirements found in the actual privacy policy.

We further believe that our formalism may aid organizations in designing their processes to avoid the possibility of or to increase the detectability of policy violations. For example, the organization can decrease violations by aligning employee incentives with the allowed purposes.

Acknowledgments. We appreciate the discussions we have had with Lorrie Faith Cranor and Joseph Y. Halpern on this work. We thank Dilsun Kaynar and Divya Sharma for many helpful comments on this paper.

References

[ABB+04] John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *Psychological Review*, 111:1036–1060, 2004.

[AF07] Sabah S. Al-Fedaghi. Beyond purpose-based privacy access control. In *ADC '07: Proceedings of the Eighteenth Conference on Australasian Database*, pages 23–32, Darlinghurst, Australia, 2007. Australian Computer Society, Inc.

[AKSX02] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Hippocratic databases. In *VLDB '02: Proceedings of the 28th International Conference on Very Large Data Bases*, pages 143–154. VLDB Endowment, 2002.

[Ans57] G.E.M. Anscombe. *Intention*. Harvard University Press, 1957.

[BA05] Travis D. Breaux and Annie I. Antón. Analyzing goal semantics for rights, permissions, and obligations. In *RE '05: Proceedings of the 13th IEEE International Conference on Requirements Engineering*, pages 177–188, Washington, DC, USA, 2005. IEEE Computer Society.

[BA08] Travis D. Breaux and Annie I. Antón. Analyzing regulatory rules for privacy and security requirements. *IEEE Trans. Softw. Eng.*, 34(1):5–20, 2008.

[Ban05] Bank of America Corporation. Bank of America privacy policy for consumers, September 2005. Accessed Feb. 4, 2011. Available from: http://www.bankofamerica.com/privacy/pdf/eng-boa.pdf.

[BBL05] Ji-Won Byun, Elisa Bertino, and Ninghui Li. Purpose based access control of complex data for privacy protection. In *SACMAT '05: Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies*, pages 102–110, New York, NY, USA, 2005. ACM.

[BKK06] Carolyn A. Brodie, Clare-Marie Karat, and John Karat. An empirical study of natural language parsing of privacy policy rules using the sparcle policy workbench. In *SOUPS '06: Proceedings of the Second Symposium on Usable Privacy and Security*, pages 8–19, New York, NY, USA, 2006. ACM.

[BKKF05] Carolyn Brodie, Clare-Marie Karat, John Karat, and Jinjuan Feng. Usable security and privacy: a case study of developing privacy management tools. In *SOUPS '05: Proceedings of the 2005 Symposium on Usable Privacy and Security*, pages 35–43, New York, NY, USA, 2005. ACM.

[BL08] Ji-Won Byun and Ninghui Li. Purpose based access control for privacy protection in relational database systems. *The VLDB Journal*, 17(4):603–619, 2008.

[BMDS07] Adam Barth, John Mitchell, Anupam Datta, and Sharada Sundaram. Privacy and utility in business processes. In *CSF '07: Proceedings of the 20th IEEE Computer Security Foundations Symposium*, pages 279–294, Washington, DC, USA, 2007. IEEE Computer Society.

[Bra87] Michael E. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, Mass., 1987.

[CMVZ09] Stephen Chong, Andrew C. Myers, K. Vikram, and Lantian Zheng. *Jif Reference Manual*, February 2009. Available from: http://www.cs.cornell.edu/jif.

[Cra02] Lorrie Faith Cranor. *Web Privacy with P3P*. O'Reilly, 2002.

[d'E63] F. d'Epenoux. A probabilistic production and inventory problem. *Management Science*, 10(1):98–108, October 1963.

[DKP96] Jagannath Prasad Das, Binod C. Kar, and Rauno K. Parrila. *Cognitive Planning: The Psychological Basis of Intelligent Behavior*. Sage, 1996.

[EKWB11] Md. Enamul Kabir, Hua Wang, and Elisa Bertino. A conditional purpose-based access control model with dynamic roles. *Expert Syst. Appl.*, 38:1482–1489, March 2011. Available from: http://dx.doi.org/10.1016/j.eswa.2010.07.057.

[Fai] FairWarning. FairWarning: Privacy breach detection for healthcare. Accessed Feb. 7, 2011. Available from: http://fairwarningaudit.com/.

[Gig02] Gerd Gigerenzer. The adaptive toolbox. In Gerd Gigerenzer and Reinhard Selten, editors, *Bounded Rationality: The Adaptive Toolbox*, Dahlem Workshop Reports, pages 37–50. MIT Press, 2002.

[GS02] Gerd Gigerenzer and Reinhard Selten, editors. *Bounded Rationality: The Adaptive Toolbox*. Dahlem Workshop Reports. MIT Press, 2002.

[HA05] Katia Hayati and Martín Abadi. Language-based enforcement of privacy policies. In *PET 2004: Workshop on Privacy Enhancing Technologies*, pages 302–313. Springer-Verlag, 2005.

[JK96] Bonnie E. John and David E. Kieras. The GOMS family of user interface analysis techniques: comparison and contrast. *ACM Trans. Comput.-Hum. Interact.*, 3:320–351, December 1996. Available from: `http://doi.acm.org/10.1145/235833.236054`.

[JSNS09] Mohammad Jafari, Reihaneh Safavi-Naini, and Nicholas Paul Sheppard. Enforcing purpose of use via workflows. In *WPES '09: Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society*, pages 113–116, New York, NY, USA, 2009. ACM. Available from: `http://doi.acm.org/10.1145/1655188.1655206`.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *STOC '84: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 302–311, New York, NY, USA, 1984. ACM. Available from: `http://doi.acm.org/10.1145/800057.808695`.

[Kha79] L. G. Khachian. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244:1093–1096, 1979. English translation in Soviet Math. Dokl. 20, 191-194, 1979.

[LDK95] Michael L. Littman, Thomas L. Dean, and Leslie P. Kaelbling. On the complexity of solving markov decision problems. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI–95)*, pages 394–402, Montreal, Québec, Canada, 1995.

[Mac74] John L. Mackie. *The Cement of the Universe: A Study of Causation*. Oxford University Press, 1974.

[MMZ06] Fabio Massacci, John Mylopoulos, and Nicola Zannone. Hierarchical hippocratic databases with minimal disclosure for virtual organizations. *The VLDB Journal*, 15(4):370–387, 2006.

[NBL+10] Qun Ni, Elisa Bertino, Jorge Lobo, Carolyn Brodie, Clare-Marie Karat, John Karat, and Alberto Trombetta. Privacy-aware role-based access control. *ACM Trans. Inf. Syst. Secur.*, 13:24:1–24:31, July 2010. Available from: `http://doi.acm.org/10.1145/1805974.1805980`.

[OED89] purpose, n. In *The Oxford English Dictionary*. Oxford University Press, 2nd edition, 1989.

[Off03] Office for Civil Rights, U.S. Department of Health and Human Services. Summary of the HIPAA privacy rule. OCR Privacy Brief, 2003.

[PGY08] Huanchun Peng, Jun Gu, and Xiaojun Ye. Dynamic purpose-based access control. In *International Symposium on Parallel and Distributed Processing with Applications*, pages 695–700, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[PS03] Calvin Powers and Matthias Schunter. Enterprise privacy authorization language (EPAL 1.2). W3C Member Submission, November 2003.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[Roy08] Olivier Roy. *Thinking before Acting: Intentions, Logic, Rational Choice*. PhD thesis, Institute for Logic, Language and Computation; Universiteit van Amsterdam, 2008.

[San96] Ravi S. Sandhu. Role hierarchies and constraints for lattice-based access controls. In *ESORICS '96: Proceedings of the 4th European Symposium on Research in Computer Security*, pages 65–79, London, UK, 1996. Springer-Verlag.

[Sel02] Reinhard Selten. What is bounded rationality? In Gerd Gigerenzer and Reinhard Selten, editors, *Bounded Rationality: The Adaptive Toolbox*, Dahlem Workshop Reports, pages 13–36. MIT Press, 2002.

[Sim55] Herbert A. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955.

[SY86] R E Strom and S Yemini. Typestate: A programming language concept for enhancing software reliability. *IEEE Trans. Softw. Eng.*, 12:157–171, January 1986. Available from: `http://portal.acm.org/citation.cfm?id=10677.10693`.

[Tay66] Richard Taylor. *Action and Purpose*. Prentice-Hall, 1966.

[The95] The European Parliament and the Council of the European Union. Directive 95/46/EC. *Official Journal of the European Union*, L 281:31–50, November 1995.

[Uni10] United States Congress. Financial services modernization act of 1999. Title 15, United States Code, Section 6802, February 2010. Accessed Feb. 4, 2011. Available from: `http://www.law.cornell.edu/uscode/15/usc_sec_15_00006802----000-.html`.

[Was03] Washington Radiology Associates, P.C. Notice of privacy practices, April 2003. Accessed Feb. 4, 2011. Available from: `http://www.washingtonradiology.com/office-guide/privacy.asp`.

[Yah10a] Yahoo! Privacy policy: Information collection and use, 2010. Available from: `http://info.yahoo.com/privacy/us/yahoo/details.html#2`.

[Yah10b] Yahoo! Privacy policy: Yahoo Mail, 2010. Available from: `http://info.yahoo.com/privacy/us/yahoo/mail/details.html`.

A Details of MDPs

One may find a discussion of MDPs in most introductions to artificial intelligence (e.g., [RN03]). For an MDP $m = \langle \mathcal{Q}, \mathcal{A}, t, r, \gamma \rangle$, the discount factor γ accounts for the preference of people for receiving rewards sooner than later. It may be thought of as similar to inflation. We require that $\gamma < 1$ to ensure that the expected total discounted reward is bounded.

The value of a state q under a strategy σ is

$$V_m(\sigma, q) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r(q_i, \sigma(q_i))\right]$$

The Bellman equation shows that

$$V_m(\sigma, q) = r(q, \sigma(q)) + \gamma \sum_{q' \in \mathcal{Q}} t(q, \sigma(q))(q') * V_m(\sigma, q')$$

A strategy σ^* is optimal if and only if for all states q, $V_m(\sigma^*, q) = \max_\sigma V_m(\sigma, q)$. At least one optimal policy always exists. Furthermore, if σ^* is optimal, then

$$\sigma^*(q) = \operatorname*{argmax}_{a \in \mathcal{A}} \left[r(s, a) + \gamma \sum_{q' \in \mathcal{Q}} t(q, \sigma(q))(q') * V_m(\sigma, q') \right]$$

B Proof of Theorem 1

The proper sub-execution relation is a strict partial order. This follows directly from the proper-subsequence relation $\sqsubset$ being a strict partial order. We write $\lhd$ for *proper sub-execution* and $\trianglelefteq$ for *proper sub-execution or equal*.

Now, we show that $\prec$ is also strict partial ordering.

- Irreflexivity: for no σ is $\sigma \prec \sigma$. For $\sigma \prec \sigma$ to be true, there would have to exist a $\sigma \in \mathsf{opt}$ such that for at least one contingency κ' and q', $m(q', \kappa', \sigma')$ is a proper sub-execution of itself. However, this is impossible since the sub-execution relation is strict partial order.

- Asymmetry: for all σ_1 and σ_2, if $\sigma_1 \prec \sigma_2$, then it is not the case that $\sigma_2 \prec \sigma_1$. To show a contradiction, suppose $\sigma_1 \prec \sigma_2$ and $\sigma_2 \prec \sigma_1$ are both true. It would have to be the case that for all contingencies κ and states q, $m(q, \kappa, \sigma_1) \trianglelefteq m(q, \kappa, \sigma_2)$ and $m(q, \kappa, \sigma_2) \trianglelefteq m(q, \kappa, \sigma_1)$. Since $\lhd$ is a strict partial order, this implies that for all q and κ, $m(q, \kappa, \sigma_1) = m(q, \kappa, \sigma_2)$. Thus, there cannot exist a contingency κ' and state q' such that $m(q', \kappa', \sigma_2) \lhd m(q', \kappa', \sigma_1)$. Then $\sigma_2 \prec \sigma_1$ cannot be true, a contradiction.

- Transitivity: for all σ_1, σ_2, and σ_3, if $\sigma_1 \prec \sigma_2$ and $\sigma_2 \prec \sigma_3$, then $\sigma_1 \prec \sigma_3$. Suppose $\sigma_1 \prec \sigma_2$ and $\sigma_2 \prec \sigma_3$. Then for all for all contingencies κ and states q, $m(q, \kappa, \sigma_1) \trianglelefteq m(q, \kappa, \sigma_2)$ and $m(q, \kappa, \sigma_2) \trianglelefteq m(q, \kappa, \sigma_3)$. Since $\trianglelefteq$ has transitivity, this implies that $m(q, \kappa, \sigma_1) \trianglelefteq m(q, \kappa, \sigma_3)$ for all κ and q.

  Furthermore, it must be the case that there exists a contingency κ' and state q' such that $m(q', \kappa', \sigma_1) \lhd m(q', \kappa', \sigma_2)$. From above, $m(q', \kappa', \sigma_2) \trianglelefteq m(q', \kappa', \sigma_3)$. Thus, by the transitivity of $\trianglelefteq$, $m(q', \kappa', \sigma_1) \lhd m(q', \kappa', \sigma_3)$ as needed.

Since $\prec$ is a strict partial ordering and $\mathcal{Q} \to \mathcal{A}$ is finite, $\mathcal{Q} \to \mathcal{A}$ is well-founded under $\prec$. $\mathcal{Q} \to \mathcal{A}$ being finite also means that $\mathsf{opt}(m)$ is finite. It is also known to be non-empty [RN03].

Suppose $\mathsf{opt}^*(m)$ were empty. This would mean for every σ of $\mathsf{opt}(m)$, there exists σ' in $\mathsf{opt}(m)$ such that $\sigma' \prec \sigma$. Since $\mathsf{opt}(m)$ is finite but non-empty, this could only happen if $\prec$ contained cycles. However, this is a contradiction since $\prec$ is a strict partial order and $\mathcal{Q} \to \mathcal{A}$ is well-founded under it. Thus, $\mathsf{opt}^*(m)$ is not empty.

C Proofs about Useless States

Proposition 1. *For all environment models m, sets U such that $U \subseteq U_m$, strategies σ, and states q, $V_m(\sigma, q) \leq V_m(U(\sigma), q)$.*

Proof. Let $\mathsf{exec}(b)$ be all the executions with the behavior b as a prefix. Let B_U be the set of all behaviors b such that for some j, $b = [q_0, a_1, q_1 \ldots, q_j, a_{j+1}, q_{j+1}]$ such that $\langle q_j, a_{j+1} \rangle$ is in U but for not $i < j$ is $\langle q_{i-1}, a_i \rangle$ in U. We may use B_U and $\mathsf{exec}(b)$ to partition the space of executions E. Thus,

$$\begin{aligned} V_m(\sigma, q) &= \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r(q_i, \sigma(q_i))\right] \\ &= \sum_{e \in E} \Pr[e|\sigma] \left[\sum_{i=0}^{\infty} \gamma^i r(q_i, \sigma(q_i))\right] \\ &= \sum_{b \in B_U} \sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \left[\sum_{i=0}^{\infty} \gamma^i r(q_i, \sigma(q_i))\right] \end{aligned} \tag{1}$$

(Note: as E is uncountable, taking a summation over it is ill advised. We could take an integral instead. Alternatively, one could take the sum over executions of bounded length. This will introduce an error term. However, as the bound increases the magnitude of this term will drop exponentially fast due to the factor γ. In essence, this is how most practical algorithms for solving MDPs operate. See [RN03].)

For any b in B_U, consider $e \in \mathsf{exec}(b)$. Since e is in $\mathsf{exec}(b)$, it must have the following form

$$[q_0, a_1, q_1 \ldots, q_j, a_{j+1}, q_{j+1}, \ldots]$$

where $\langle q_j, a_{j+1} \rangle \in U$ but for $i < j$ is $\langle q_i, a_{i+1} \rangle \notin U$ where $b = [q_0, a_1, q_1 \ldots, q_j, a_{j+1}, q_{j+1}]$.

For $\sigma \in \mathsf{strg}(b)$, we reason as shown as follows.

$$\begin{aligned}
&\sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \left[\sum_{i=0}^{\infty} \gamma^i r(q_i, \sigma(q_i)) \right] \\
&\qquad = \sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \left[\sum_{i=0}^{j-1} \gamma^i r(q_i, \sigma(q_i)) + \sum_{i=j}^{\infty} \gamma^i r(q_i, \sigma(q_i)) \right] \\
&\qquad = \left[\sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \sum_{i=0}^{j-1} \gamma^i r(q_i, \sigma(q_i)) \right] + \gamma^j \left[\sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \sum_{i=j}^{\infty} \gamma^{i-j} r(q_i, \sigma(q_i)) \right] \\
&\qquad = \left[\sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \sum_{i=0}^{j-1} \gamma^i r(q_i, \sigma(q_i)) \right] + \gamma^j \mathbb{E} \left[\sum_{i=j}^{\infty} \gamma^i r(q_i, \sigma(q_i)) \right] \\
&\qquad = \left[\sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \sum_{i=0}^{j-1} \gamma^i r(q_i, \sigma(q_i)) \right] + \gamma^j V_m(\sigma, q_j) \qquad (2)
\end{aligned}$$

Furthermore,

$$\sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \sum_{i=0}^{j-1} \gamma^i r(q_i, \sigma(q_i)) = \Pr[b|\sigma] \sum_{i=0}^{j-1} \gamma^i r(q_i, \sigma(q_i))$$

Thus, the left term is equal under σ and $U(\sigma)$:

$$\sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \sum_{i=0}^{j-1} \gamma^i r(q_i, \sigma(q_i)) = \Pr[b|\sigma] \sum_{i=0}^{j-1} \gamma^i r(q_i, \sigma(q_i)) \qquad (3)$$

$$= \Pr[b|\sigma] \sum_{i=0}^{j-1} \gamma^i r(q_i, U(\sigma)(q_i)) \qquad (4)$$

$$= \sum_{e \in \mathsf{exec}(b)} \Pr[e|U(\sigma)] \sum_{i=0}^{j-1} \gamma^i r(q_i, U(\sigma)(q_i)) \qquad (5)$$

where line 4 follows since $\sigma(q_i) = U(\sigma)(q_i)$ for $\langle q_i, a_{i+1} \rangle \notin U$.

Since $\langle q_j, a_{j+1} \rangle \in U$, we know that $Q_m(\sigma, q_j, a_{j+1}) \leq 0$. Furthermore, since $\sigma \in \mathsf{strg}(b)$, it is the case that $\sigma(q_j) = a_{j+1}$. Thus, $V_m(\sigma, q_j) = Q_m(\sigma, q_j, \sigma(q_j)) \leq 0$. Furthermore, since $\langle q_j, a_{j+1} \rangle \in U$,

$V_m(U(\sigma), q_j) = Q_m(\sigma, q_j, \mathsf{N}) = 0$. Thus, we may conclude

$$\sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \left[\sum_{i=0}^{\infty} \gamma^i r(q_i, \sigma(q_i)) \right]$$

$$= \left[\sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \sum_{i=0}^{j-1} \gamma^i r(q_i, \sigma(q_i)) \right] + \gamma^j V_m(\sigma, q_j) \tag{6}$$

$$= \left[\sum_{e \in \mathsf{exec}(b)} \Pr[e|U(\sigma)] \sum_{i=0}^{j-1} \gamma^i r(q_i, U(\sigma)(q_i)) \right] + \gamma^j V_m(\sigma, q_j) \tag{7}$$

$$\leq \left[\sum_{e \in \mathsf{exec}(b)} \Pr[e|U(\sigma)] \sum_{i=0}^{j-1} \gamma^i r(q_i, U(\sigma(q_i))) \right] + \gamma^j V_m(U(\sigma), q_j) \tag{8}$$

(9)

$$= \sum_{e \in \mathsf{exec}(b)} \Pr[e|U(\sigma)] \left[\sum_{i=0}^{\infty} \gamma^i r(q_i, U(\sigma)(q_i)) \right] \tag{10}$$

where lines 6 and 10 come from the reasoning leading to line 2, and line 7 comes from the reasoning leading to line 5.

Note that the above also trivially holds when $\sigma \notin \mathsf{strg}(b)$ since $\Pr[e|\sigma] = 0$ and $\Pr[e|U(\sigma)] = 0$ for all $e \in \mathsf{exec}(b)$. Thus, for all σ, we have

$$\sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \left[\sum_{i=0}^{\infty} \gamma^i r(q_i, \sigma(q_i)) \right] \leq \sum_{e \in \mathsf{exec}(b)} \Pr[e|U(\sigma)] \left[\sum_{i=0}^{\infty} \gamma^i r(q_i, U(\sigma)(q_i)) \right] \tag{11}$$

Thus,

$$V_m(\sigma, q) = \sum_{b \in B_U} \sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \left[\sum_{i=0}^{\infty} \gamma^i r(q_i, \sigma(q_i)) \right] \tag{12}$$

$$\leq \sum_{b \in B_U} \sum_{e \in \mathsf{exec}(b)} \Pr[e|\sigma] \left[\sum_{i=0}^{\infty} \gamma^i r(q_i, \sigma(q_i)) \right] \tag{13}$$

$$= V_m(U(\sigma), q) \tag{14}$$

where line 12 and 14 comes from the reasoning of line 1, and line 13 comes from equation 11. □

D Proof of Lemma 1

First we prove that this $\log^{-1}(b) \cap \mathsf{behv}^*(m)$ in the lemma may be replaced with $\mathsf{strg}_m(b) \cap \mathsf{opt}^*(m)$. Then, we prove the modified statement with two propositions. We have one corresponding to the *if* direction and one to the *only if* direction.

Proposition 2. *For environment models m, if for all observable behaviors b, $\log(b) = b$, then $\mathsf{strg}(b) \cap \mathsf{opt}^*(m)$ is empty if and only if $\log^{-1}(b) \cap \mathsf{behv}^*(m)$ is empty.*

Proof. Since $\log^{-1}(b) = \{b\}$, $\log^{-1}(b) \cap \mathsf{behv}^*(m)$ is empty if and only $b \notin \mathsf{behv}^*(m)$. b is in $\mathsf{behv}^*(m)$ if and only if there exists a strategy σ in $\mathsf{opt}^*(m)$ such that there exists a contingency κ, and a state q such that b is a subsequence of $m(q, \kappa, \sigma)$.

For all σ in $\mathsf{opt}^*(m)$, $\exists \kappa, q.b \sqsubset m(q, \kappa, \sigma)$ is equivalent to $\forall i \in [0, n).\sigma(q_i) = a_{i+1}$ where $b = [q_0, a_1, q_1, a_2, \ldots, a_n, q_n]$. To see this, note b was observed and, thus, it must have been produced by a contingency consistent with m.

$\forall i \in [0, n).\sigma(q_i) = a_{i+1}$ is equivalent to $\sigma \in \mathsf{strg}(b)$. Thus, b is in $\mathsf{behv}^*(m)$ if and only if there exists a strategy σ in $\mathsf{opt}^*(m)$ such that σ is in $\mathsf{strg}(b)$. Thus, $\log^{-1}(b) \cap \mathsf{behv}^*(m)$ is not empty if and only if $\mathsf{strg}(b) \cap \mathsf{opt}^*(m)$ is not empty. □

Proposition 3. *For all environment models m and behaviors $b = [q_0, a_1, q_1, \ldots, a_n, q_n]$, $\mathsf{strg}(b) \cap \mathsf{opt}^*(m)$ is not empty if (1) for all i such that $0 \leq i < n$, $\langle q_i, a_{i+1} \rangle \notin U_m$ and (2) $\mathsf{strg}(b) \cap \mathsf{opt}(m)$ is not empty.*

Proof. Suppose the conditions (1) and (2) are true. Since $\mathsf{strg}(b) \cap \mathsf{opt}(m)$ is not empty, there exists some σ_1 in both of them. Since σ_1 is in $\mathsf{strg}(b)$, for all $0 \leq i < n$, $\sigma_1(q_i) = a_{i+1}$. Thus, by condition (2), $\langle q_i, \sigma_1(q_i) \rangle \notin U_m$. This further implies that a_{i+1} is not N.

Let $\sigma_2 = U_m(\sigma_1)$. σ_2 is in $\mathsf{strg}(b)$ because for all $0 \leq i < n$, $\sigma_1(q_i) = \sigma_2(q_i)$ since $\langle q_i, \sigma_1(q_i) \rangle \notin U_m$. Furthermore, by Proposition 1, for all q, $V_m(\sigma_1, q) \leq V_m(\sigma_2, q)$. Thus, σ_2 is in $\mathsf{opt}(m)$ as well.

To show that σ_2 is also in $\mathsf{opt}^*(m)$, suppose it were not. Since σ_2 is in $\mathsf{opt}(m)$, this implies that there exists σ' in $\mathsf{opt}(m)$ such that $\sigma' \prec \sigma_2$. For this to be true, there must exist κ' and state q' such that $\mathsf{active}(m(q', \kappa', \sigma')) \sqsubset \mathsf{active}(m(q', \kappa', \sigma_2)$. Thus, for some i, $m(q', \kappa', \sigma_2)$ must have the form $[q_0, a_1, q_1, \ldots, q_{i-1}, a_i, q_i, a_{i+1}, q_{i+1}, \ldots]$, and $m(q', \kappa', \sigma')$ must have the form $[q_0, a_1, q_1, \ldots, q_{i-1}, a_i, q_i, \mathsf{N}, q_i, \ldots]$ where a_{i+1} is not N. Since $\sigma_2(q_i) = a_{i+1}$, by the construction of σ_2, $\langle q_i, a_{i+1} \rangle$ is not in U_m. Thus, there exists some σ_3 such that $Q_m(\sigma_3, q_i, a_{i+1}) > 0$.

Since σ_2 is in $\mathsf{opt}(m)$, $Q_m(\sigma_2, q_i, a_{i+1}) \geq Q_m(\sigma_3, q_i, a_{i+1}) > 0$. Thus, we have $V_m(\sigma_2, q_i) = Q_m(\sigma_2, q_i, a_{i+1}) > 0$. However, $V_m(\sigma', q_i) = 0$ meaning that σ' is not in $\mathsf{opt}(m)$, a contradiction. □

Proposition 4. *For all environment models m and behaviors $b = [q_0, a_1, q_1, \ldots, a_n, q_n]$, if $\mathsf{strg}(b) \cap \mathsf{opt}^*(m)$ is not empty, then (1) for all i such that $0 \leq i < n$, $\langle q_i, a_{i+1} \rangle \notin U_m$ and (2) $\mathsf{strg}(b) \cap \mathsf{opt}(m)$ is not empty.*

Proof. Condition (2) follows from the fact that $\mathsf{opt}^*(m) \subseteq \mathsf{opt}(m)$.

To prove condition (1), suppose $\mathsf{strg}(b) \cap \mathsf{opt}^*(m)$ is not empty but condition (1) does not hold. Then there exists σ_1 in $\mathsf{strg}(b) \cap \mathsf{opt}^*(m)$. Furthermore, there exists some i' such that $\langle q_{i'}, a_{i'+1} \rangle \in U_m$. Since $\sigma_1 \in \mathsf{strg}(b)$, it must be the case that for all $i < n$, $a_{i+1} = \sigma(q_i)$. Thus, $\sigma_1(q_{i'}) = a_{i'+1}$. By Proposition 1, for all q, $V_m(\sigma_1, q) \leq V_m(U_m(\sigma_1), q)$. Furthermore, $U(\sigma_1) \prec \sigma_1$. To see this, recall that U_m is not empty. Thus, any contingency κ' that results in state $q_{i'}$, $m(q_0, \kappa', U_m(\sigma_1)) \sqsubset m(q_0, \kappa', \sigma_1)$ since only $U_m(\sigma_1)$ does nothing at $q_{i'}$. For κ that do not lead to $q_{i'}$, the two executions will be the same.

Since $U_m(\sigma_1) \prec \sigma_1$ and $U(\sigma_1)$ is in $\mathsf{opt}(m)$, σ_1 cannot be in $\mathsf{opt}^*(b)$, a contradiction. □

E Proof of Lemma 2

If $\langle q, a \rangle$ is in U_m, then $a \neq \mathsf{N}$ and for all strategies σ, $Q_m(\sigma, q, a) \leq 0$. Thus, the lemma is true if the following is true: $Q^*(q, a) \leq 0$ iff $\forall \sigma.Q_m(\sigma, q, a) \leq 0$.

To show this, note that $\forall\sigma.Q_m(\sigma,q,a) \leq 0$ iff $\max_\sigma Q_m(\sigma,q,a) \leq 0$. Furthermore,

$$\begin{aligned}
\max_\sigma Q_m(\sigma,q,a) &= \max_\sigma r(q,a) + \gamma \sum_{q'} t(q,a)(q') * V_m(\sigma,q') \\
&= r(q,a) + \gamma \sum_{q'} t(q,a)(q') * \max_\sigma V_m(\sigma,q') \\
&= r(q,a) + \gamma \sum_{q'} t(q,a)(q') * V^*(q') \\
&= Q^*_m(q,a)
\end{aligned}$$

Thus, $\forall\sigma.Q_m(\sigma,q,a) \leq 0$ iff $Q^*_m(q,a) \leq 0$.

F Properties of fix

Proposition 5. *For all environment models m, strategies σ, and states q, $V_{\mathsf{fix}(m,b)}(\sigma,q) \leq V_m(\sigma,q)$.*

Proof. Let $m = \langle \mathcal{Q}, \mathcal{A}, t, r, \gamma\rangle$ and $\mathsf{fix}(m,b) = \langle \mathcal{Q}, \mathcal{A}, t, r', \gamma\rangle$.

$$V_m(\sigma,q) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r(q_i, \sigma(q_i))\right] \tag{15}$$

$$\leq \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r'(q_i, \sigma(q_i))\right] \tag{16}$$

$$= V_{\mathsf{fix}(m,b)}(\sigma,q) \tag{17}$$

where line 16 follows from the fact that for all q and a, $r'(q,a) \leq r(q,a)$. □

Proposition 6. *For all environment models m, behaviors b, $\sigma \in \mathsf{strg}(b)$, and states q, $V_{\mathsf{fix}(m,b)}(\sigma,q) = V_m(\sigma,q)$*

Proof. Let $m = \langle \mathcal{Q}, \mathcal{A}, t, r, \gamma\rangle$ and $\mathsf{fix}(m,b) = \langle \mathcal{Q}, \mathcal{A}, t, r', \gamma\rangle$. Let $b = [q_0, a_1, q_1, \ldots, a_n, q_n]$.

Since σ is in $\mathsf{strg}(b)$, for all i such that $0 \leq i < n$, $\sigma(q_i) = a_{i+1}$. Thus, $r'(q_i, a_{i+1}) = r(q_i, a_{i+1})$. For all q that is not equal to q_i for any i, $r'(q,a) = r(q,a)$ for all a. Thus, for all a and q, $r'(q,\sigma(q)) = r(q,\sigma(q))$. This implies

$$\begin{aligned}
V_m(\sigma,q) &= \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r(q_i, \sigma(q_i))\right] \\
&= \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r'(q_i, \sigma(q_i))\right] \\
&= V_{\mathsf{fix}(m,b)}(\sigma,q)
\end{aligned}$$

□

Proposition 7. *For all environment models m, behaviors b, and $\sigma_1 \notin \mathsf{strg}(b)$, there exists a $\sigma_2 \in \mathsf{strg}(b)$ such that for all states q, $V_{\mathsf{fix}(m,b)}(\sigma_1,q) \leq V_{\mathsf{fix}(m,b)}(\sigma_2,q)$.*

Proof. Let $\mathsf{fix}(m,b) = \langle \mathcal{Q}, \mathcal{A}, t, r', \gamma \rangle$. Let $b = [q_0, a_1, q_1, \ldots, a_n, q_n]$. Since σ_1 is not in $\mathsf{strg}(b)$, there must exist some i such that $\sigma_1(q_i) \neq a_{i+1}$. Let the set I hold all such indexes i: $I = \{\, i \in [0,n) \mid \sigma_1(q_i) \neq a_{i+1} \,\}$. Let σ_2 be the strategy such that $\sigma_2(q) = a_{i+1}$ if $q = q_i$ for some $i \in I$ and $\sigma_2(q) = \sigma_1(q)$ otherwise. By construction, σ_2 is in $\mathsf{strg}(b)$.

By the construction of $\mathsf{fix}(m,b)$, for all $i \in I$, $r'(q_i, \sigma_1(q_i)) = -\omega \leq r'(q_i, a_{i+1}) = r'(q_i, \sigma_2(q_i))$. Thus, for all q, $r'(q, \sigma_1(q)) \leq r'(q, \sigma_2(q))$. Thus, for all states q, $V_{\mathsf{fix}(m,b)}(\sigma_1, q) \leq V_{\mathsf{fix}(m,b)}(\sigma_2, q)$. □

Proposition 8. *For all environment models m, behaviors b, $\sigma_1 \notin \mathsf{strg}(b)$, and $\sigma_2 \in \mathsf{strg}(b)$, there exists a state q such that $V_{\mathsf{fix}(m,b)}(\sigma_1, q) < V_{\mathsf{fix}(m,b)}(\sigma_2, q)$.*

Proof. Let $b = [q_0, a_1, q_1, \ldots, a_n, q_n]$. Since σ_1 is not in $\mathsf{strg}(b)$, there must exist some i such that $\sigma_1(q_i) \neq a_{i+1}$. By the construction of $\mathsf{fix}(m,b)$, $r'(q_i, \sigma_1(q_i)) = -\omega$. Recall that $\omega > 2r^*/(1-\gamma)$ where r^* is the reward with the largest magnitude. Thus,

$$V_{\mathsf{fix}(m,b)}(\sigma_1, q_i) = r(q_i, \sigma_1(q_i)) + \gamma \sum_{q'} t(q_i, \sigma_1(q_i))(q') * V_m(\sigma, q') \tag{18}$$

$$= -\omega + \gamma \sum_{q'} t(q_i, \sigma_1(q_i))(q') * V_m(\sigma, q') \tag{19}$$

$$\leq -\omega + \gamma \sum_{q'} t(q_i, \sigma_1(q_i))(q') * r^*/(1-\gamma) \tag{20}$$

$$= -\omega + \gamma * r^*/(1-\gamma) \tag{21}$$

$$\leq -\omega + r^*/(1-\gamma) \tag{22}$$

$$< -[2r^*/(1-\gamma)] + r^*/(1-\gamma) \tag{23}$$

$$= -r^*/(1-\gamma) \tag{24}$$

$$\leq V_m(\sigma_2, q) \tag{25}$$

$$= V_{\mathsf{fix}(m,b)}(\sigma_2, q) \tag{26}$$

where line 21 follows from $t(q, \sigma_1(q))$ being a probability distribution over states, line 25 follows from the definition of r^* and known bounds (e.g., [RN03]), and line 26 follows from by Proposition 6. □

Proposition 9. *For all environment models m and behaviors b, $\mathsf{opt}(\mathsf{fix}(m,b))$ is a subset of $\mathsf{strg}(b)$.*

Proof. Suppose σ_1 were not in $\mathsf{strg}(b)$. By Proposition 8, for all $\sigma_2 \in \mathsf{strg}(b)$, there exists a state q such that $V_{\mathsf{fix}(m,b)}(\sigma_1, q) < V_{\mathsf{fix}(m,b)}(\sigma_2, q)$. Thus, σ_1 is not in $\mathsf{opt}(\mathsf{fix}(m,b))$. □

Proposition 10. *For all environment models m, behaviors b, and strategies σ in $\mathsf{opt}(\mathsf{fix}(m,b))$, $V_{\mathsf{fix}(m,b)}(\sigma, q) = V_m(\sigma, q)$.*

Proof. Let σ be in $\mathsf{opt}(\mathsf{fix}(m,b))$. σ must be in $\mathsf{strg}(m,b)$ by Proposition 9. Thus, $V_{\mathsf{fix}(m,b)}(\sigma, q) = V_m(\sigma, q)$ by Proposition 6. □

G Proof of Lemma 3

This lemma follows directly from the Propositions 11 and 12 below.

Proposition 11. *For all environment models m and behaviors b, $\mathsf{strg}(b) \cap \mathsf{opt}(m) = \mathsf{opt}(\mathsf{fix}(m,b)) \cap \mathsf{opt}(m)$.*

Proof. Consider the set $\mathsf{strg\text{-}opt}(m,b) = \mathsf{strg}(b) - \mathsf{opt}(\mathsf{fix}(m,b))$. For all σ in $\mathsf{strg\text{-}opt}(m,b)$, σ is in $\mathsf{strg}(b)$ but not $\mathsf{opt}(\mathsf{fix}(m,b))$. By being in $\mathsf{strg}(b)$, $V_{\mathsf{fix}(m,b)}(\sigma,q) = V_m(\sigma,q)$ by Proposition 6. Thus, since σ is not in $\mathsf{opt}(\mathsf{fix}(m,b))$, σ is not in $\mathsf{opt}(m)$ either by Proposition 5. This means that $\mathsf{strg\text{-}opt}(m,b) \cap \mathsf{opt}(m)$ is empty.

Furthermore, $\mathsf{opt}(\mathsf{fix}(m,b)) \subseteq \mathsf{strg}(b)$ by Proposition 9. Thus, $\mathsf{strg}(b) = \mathsf{opt}(\mathsf{fix}(m,b)) \cup (\mathsf{strg}(b) - \mathsf{opt}(\mathsf{fix}(m,b))) = \mathsf{opt}(\mathsf{fix}(m,b)) \cup \mathsf{strg\text{-}opt}(m,b)$. Thus,

$$\begin{aligned}\mathsf{strg}(b) \cap \mathsf{opt}(m) &= (\mathsf{opt}(\mathsf{fix}(m,b)) \cup \mathsf{strg\text{-}opt}(m,b)) \cap \mathsf{opt}(m) \\ &= (\mathsf{opt}(\mathsf{fix}(m,b)) \cap \mathsf{opt}(m)) \cup (\mathsf{strg\text{-}opt}(m,b) \cap \mathsf{opt}(m)) \\ &= \mathsf{opt}(\mathsf{fix}(m,b)) \cap \mathsf{opt}(m)\end{aligned}$$

□

Proposition 12. *For all environment models m and behaviors b, $\mathsf{opt}(m) \cap \mathsf{opt}(\mathsf{fix}(m,b))$ is empty if and only if for all q, $\max_\sigma V_{\mathsf{fix}(m,b)}(\sigma,q) \neq \max_\sigma V_m(\sigma,q)$.*

Proof. Suppose that $\mathsf{opt}(m) \cap \mathsf{opt}(\mathsf{fix}(m,b))$ is not empty. Then there exists σ^* in both of them. Thus,

$$\max_\sigma V_{\mathsf{fix}(m,b)}(\sigma,q) = V_{\mathsf{fix}(m,b)}(\sigma^*,q) \tag{27}$$

$$= V_m(\sigma^*,q) \tag{28}$$

$$= \max_\sigma V_m(\sigma,q) \tag{29}$$

where line 28 follows from Proposition 10 and lines 27 and 29 follow from σ^* being in both $\mathsf{opt}(m)$ and $\mathsf{opt}(\mathsf{fix}(m,b))$.

Suppose that for all q, $\max_\sigma V_m(\sigma,q_0) = \max_\sigma V_{\mathsf{fix}(m,b)}(\sigma,q_0)$. Let σ^* be in $\mathsf{opt}(\mathsf{fix}(m,b))$. For all q,

$$V_m(\sigma^*,q) = V_{\mathsf{fix}(m,b)}(\sigma^*,q) \tag{30}$$

$$= \max_\sigma V_{\mathsf{fix}(m,b)}(\sigma,q_0) \tag{31}$$

$$= \max_\sigma V_m(\sigma,q) \tag{32}$$

where line 30 follows from Proposition 10 and line 31 from $\sigma^* \in \mathsf{opt}(\mathsf{fix}(m,b))$. Thus, σ^* is in $\mathsf{opt}(m)$ and $\mathsf{opt}(m) \cap \mathsf{opt}(\mathsf{fix}(m,b))$ is not empty. □

H Proof of Theorem 2

Line 05 will return *true* if there exists i such that $a_{i+1} \neq \mathsf{N}$ and $Q^*_m(q_i,a_{i+1}) \leq 0$. By Lemma 2, this implies that $\langle q_i, a_{i+1}$ is in U_m. By Lemma 1, this implies that $\mathsf{log}^{-1}(b) \cap \mathsf{behv}^*(m)$ is empty under condition (1).

Lines 06–16 constructs $m' = \mathsf{fix}(m,b)$. It constructs r' from r by first setting $r' = r$. On lines 13–16, it then sets $r'(q_i,k)$ to be $-\omega$ for all k such that $k \neq a_{i+1}$. Thus, $r'(q_i,a_{i+1})$ will be left as $r(q_i,a_{i+1})$ as needed.

If Line 05 does not Line 21 will return *false* if there exists j such that $V^*_m(j) = V^*_m(j)$. In this case, it cannot be that for all q, $\max_\sigma V_m(\sigma, q_0) = \max_\sigma V_{\mathsf{fix}(m,b)}(\sigma, q_0)$. Thus, by Lemma 3, $\mathsf{strg}(b) \cap \mathsf{opt}(m)$ is not empty and condition (2) is false of Lemma 1. Since the function would had returned already at Line 05 if condition (1) were true, we know it is false. Thus, by Lemma 1, $\mathsf{log}^{-1}(b) \cap \mathsf{behv}^*(m)$ is not empty.

If Line 22 is reached, *true* is returned. This can only happen if condition (2) is true. This implies that $\mathsf{log}^{-1}(b) \cap \mathsf{behv}^*(m)$ is empty by Lemma 1.

Thus, the algorithm is correct whether it returns *true* or *false*.