Formalizing and Enforcing Purpose Restrictions in Privacy Policies

Michael Carl. Tschantz  
Carnegie Mellon University

Anupam Datta  
Carnegie Mellon University

Jeannette M. Wing  
Carnegie Mellon University

Follow this and additional works at: http://repository.cmu.edu/compsci

Published In
Formalizing and Enforcing Purpose Restrictions in Privacy Policies

Michael Carl Tschantz  
Carnegie Mellon University  
Email: mtschant@cs.cmu.edu

Anupam Datta  
Carnegie Mellon University  
Email: danupam@cmu.edu

Jeannette M. Wing  
Carnegie Mellon University  
Email: wing@cs.cmu.edu

Abstract—Privacy policies often place restrictions on the purposes for which a governed entity may use personal information. For example, regulations, such as the Health Insurance Portability and Accountability Act (HIPAA), require that hospital employees use medical information for only certain purposes, such as treatment, but not for others, such as gossip. Thus, using formal or automated methods for enforcing privacy policies requires a semantics of purpose restrictions 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 (MDPs), which exclude redundant actions for a formal definition of redundant. We argue that an action is for a purpose if and only if the action is part of a plan for optimizing the satisfaction of that purpose under the MDP model. We use this formalization to define when a sequence of actions is only for or not for a purpose. This semantics enables us to create and implement an algorithm for automating auditing, and to describe formally and compare rigorously previous enforcement methods. To validate our semantics, we conduct a survey to compare our semantics to how people commonly understand the word “purpose”.

I. INTRODUCTION

Purpose is a key concept for privacy policies. For example, the European Union requires that [1]:

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 restrictions on information in some domains such as the Health Insurance Portability and Accountability Act (HIPAA) [2] for medical information and the Gramm-Leach-Bliley Act [3] 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 […]” [4]. 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” [5, 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.” [6, emphasis added]. The HIPAA Privacy Rule requires that health care providers only use protected health information about a patient with that patient’s authorization or for a fixed list of allowed purposes, such as treatment and billing [2].

These examples show that verifying that an organization obeys a privacy policy requires a semantics of purpose restrictions. In particular, enforcement requires the ability to determine that the organization obeys at least two classes of purpose restrictions. Yahoo!’s privacy policy shows an example of the first class: a rule requiring that an organization does not use certain information for a purpose. HIPAA provides an example of the second class: a rule requiring that an organization use certain information only for a given list of purposes. We call the first class of restrictions prohibitive rules (not-for) and the second class exclusivity rules (only-for). A prohibitive rule disallows an action for a particular purpose. An exclusivity rule disallows an action for every purpose other than the exceptions the rule lists. Each class of rule requires determining whether the organization’s behavior is for a purpose, but they differ in whether this determination indicates a violation or compliance.

Manual enforcement of privacy policies is labor intensive and error prone [7]. Thus, to reduce costs and build trust, organizations should automate the enforcement of their privacy policies; tool support for this activity is emerging in the market. For example, Fair Warning sells automated services to hospitals for detecting privacy breaches [7]. Meanwhile, previous research has proposed formal methods to enforce purpose restrictions [8]–[15].

However, each of these endeavors starts 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 whether an action could be for a purpose. It would also show which enforcement methods are most accurate. More fundamentally, it could frame the scientific basis of a societal and legal understanding of purpose and of privacy policies. Such a foundation can, for example, guide implementers as they codify in software an organization’s privacy policies.

The goal of this work is to study the meaning of purpose in the context of enforcing privacy policies. We aim to provide formal definitions suitable for automating the enforcement of purpose restrictions. We focus on automated auditing since we find that post-hoc auditing by a trusted auditor provides the perspective often required to determine the purpose of an action. However, we believe our semantics is applicable to other enforcement mechanisms and may also clarify informal reasoning. For example, in Section V-C, we use it to create an operating procedure that encourages compliance with a purpose restriction.

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 [16]:

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” [17, p. 19]. In Section II, we present an example making this relationship between planning and purpose explicit. We (as have philosophers [18]) conclude that if an auditee (the person or organization being audited) chooses to perform an action a while planning to achieve the purpose p, then the auditee’s action a is for the purpose p. Our goal is to make these notions formal in a manner useful for the automation of auditing.

In Section III, we present a formalism based upon these intuitions. We formalize planning using Markov Decision Processes (MDPs) and provide semantics to purpose restrictions based upon planning with MDPs. Section IV provides an auditing method and discusses the ramifications of the auditor observing only the behaviors of the auditee and not the underlying planning process of the auditee that resulted in these behaviors. We characterize circumstances in which 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 V presents an implemented algorithm for auditing based on our formal definitions and also shows how to use it to create an operating procedure that encourages compliance with a purpose restriction.

To validate our semantics, we perform an empirical study. In Section VI, we present the results of a survey testing how people understand the word “purpose”. The survey compares our planning based method to the prior method based on whether an action improves the satisfaction of a purpose. We find that our method matches the survey participants’ responses much more closely than the prior method.

In Section VII, we use our formalism to discuss the strengths and weaknesses of each previous method. In particular, we find that each method enforces the policy given the set of all possible allowed behaviors, which is a set that our method can construct. We also compare the previous auditing methods, which differ in their trade-offs between auditing complexity and accuracy of representing this set of behaviors. Section VIII discusses other related work.

Our work makes the following contributions:

1) The first semantic formalism of when a sequence of actions is for a purpose;
2) Empirical validation that our formalism closely corresponds to how people understand the word “purpose”;
3) An algorithm employing our formalism and its implementation for auditing; and
4) The characterization of previous policy enforcement methods in our formalism and a comparative study of their expressiveness.

The first two contributions illustrate that planning can formalize purpose restrictions. The next two illustrate that our formalism may aid automated auditing and analysis. While we view these results as a significant step towards enforcement of practical privacy policies with purpose restrictions, we recognize that further work is needed before we will have audit tools that are ready for use in organizations that must comply with complex policies. We outline concrete directions for future work towards this goal in Section IX.

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 or statements of ethics proscribing conflicts of interest.

A related technical report offers proofs and additional details [19].

II. 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 possible the alternate motivation of profit.

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 test 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 agent takes an action 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, to reach a diagnosis, must either send the medical record to a specialist or take an MRI. In this scenario, the physician’s sending the record to the specialist is not necessary since he could take an MRI. Likewise, taking the MRI is not necessary. Yet, the physician must do one or the other and that action 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., [20]). 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 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 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 either the specialist or an MRI 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 the total purpose score accumulated over all his actions.

We must adjust our notion of non-redundancy accordingly. An action is non-redundant if removing that action from the sequence would result in the purpose being satisfied less. Now, even if the physician can make a diagnosis himself, sending the record to a specialist would be non-redundant if getting a second opinion improves the quality of the diagnosis.

**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 outcome of the physician’s actions 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.

III. PLANNING FOR A PURPOSE

Now, 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 planning with 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 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.

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.

A. Markov Decision Processes

An MDP may be thought of as a probabilistic automaton where each transition is labeled with a reward in addition to an action. Rather than having accepting or goal states, the “goal” of an MDP is to maximize the total reward over time.

An MDP is a tuple \( m = (S, A, t, r, \gamma) \) where

- \( S \) is a finite set of states;
- \( A \) is a finite set of actions;
- \( t : S \times A \rightarrow D(S) \), a transition function from a state and an action to a distribution over states (represented as \( D(S) \));
- \( r : S \times A \rightarrow \mathbb{R} \), a reward function; and
- \( \gamma \), a discount factor such that \( 0 < \gamma < 1 \).

For each state \( s \) in \( S \), the agent using the MDP to plan selects an action \( a \) from \( A \) to perform. Upon performing the action \( a \) in the state \( s \), the agent receives the reward \( r(s, a) \). The environment then transitions to a new state \( s' \) with probability \( \mu(s') \) where \( \mu \) is the distribution provided by \( t(s, a) \). The goal of the agent is to select actions to maximize its expected total discounted reward \( E \left[ \sum_{i=0}^{\infty} \gamma^i \rho_i \right] \) where \( i \in \mathbb{N} \) (the set of natural numbers) ranges over time modeled as discrete steps, \( \rho_i \) is the reward at time \( i \), and the expectation is taken over the probabilistic transitions. The discount factor \( \gamma \) accounts for the preference of people to receive 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.

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 \( \sigma \) from the state space \( S \) to the set \( A \) of actions (i.e., \( \sigma : S \rightarrow A \)) such that at a state \( s \) in \( S \), the agent always selects to perform the action \( \sigma(s) \). The value of a state \( s \) under a strategy \( \sigma \) is \( V_m(\sigma, s) = E \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, \sigma(s_i)) \right] \). The Bellman equation [21] shows that

\[
V_m(\sigma, s) = r(s, \sigma(s)) + \gamma \sum_{s' \in S} t(s, \sigma(s))(s') \ast V_m(\sigma, s')
\]

A strategy \( \sigma^* \) is optimal if and only if for all states \( s \), \( V_m(\sigma^*, s) = \max_{\sigma} V_m(\sigma, s) \). At least one optimal policy always exists (see, e.g., [22]). Furthermore, if \( \sigma^* \) is optimal, then

\[
\sigma^*(s) = \arg\max_{a \in A} \left[ r(s, a) + \gamma \sum_{s' \in S} t(s, \sigma(s))(s') \ast V_m(\sigma, s') \right]
\]

We denote this set of optimal strategies as \( \text{opt}(\beta, A, t, r, \gamma) \), or when the transition system is clear from context, as \( \text{opt}(r) \). Such strategies are sufficient to maximize the agent’s expected total discounted reward despite depending only upon the current state of the MDP.

Given the strategy \( \sigma \) and the actual results of the probabilistic transitions yielded by \( t \), the agent exhibits an execution. We represent this execution as an infinite sequence \( e = [s_1, a_1, s_2, a_2, \ldots] \) of alternating states and actions starting with a state, where \( s_i \) is the \( i \)th state that the agent was in and \( a_i \) is the \( i \)th action the agent took, for all \( i \in \mathbb{N} \). We say an execution \( e \) is consistent with a strategy \( \sigma \) if and only if \( a_i = \sigma(s_i) \) for all \( i \) in \( \mathbb{N} \) where \( a_i \) is the \( i \)th action in \( e \) and \( s_i \) is the \( i \)th 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 degree, 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 \( r^p(s, a) \) is the degree to which taking the action \( a \) from state \( s \) 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 \( \sigma \) that is optimal for the MDP \( (S, A, t, r^p, \gamma) \).

B. Non-redundancy

MDPs do not require that strategies be non-redundant. Even given that the auditee had an execution \( e \) from using a
strategy \( \sigma \) in \( \text{opt}(r^p) \), 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 \( \text{opt} \) 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 \( \text{opt} \) with a new optimization criterion \( \text{nopt} \) 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 \( \text{Stop} \) that stands for stopping and doing nothing more. For all states \( s \), \( \text{Stop} \) labels a transition with zero reward (i.e., \( r(s, \text{Stop}) = 0 \)) that is a self-loop (i.e., \( t(s, \text{Stop})(s) = 1 \)). (We could put \( \text{Stop} \) on only the subset of states that represent possible stopping points by slightly complicating our formalism.) Since we only allow deterministic stationary strategies and \( \text{Stop} \) only labels self-loops, this decision is irrevocable: once the agent stops and does nothing, it does nothing 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 \( \text{active}(e) \) denote the prefix of \( e \) before the first instance of the nothing actions. \( \text{active}(e) \) will be equal to \( e \) in the case where \( e \) does not contain the nothing action.

We use the idea of doing nothing to make formal when one execution contains 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 \( \text{active}(e_1) \) is a proper prefix of \( \text{active}(e_2) \) using the standard notion of prefix. 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 \( \kappa \) be a function from \( S \times A \times \mathbb{N} \) to \( S \) such that \( \kappa(s, a, i) \) is the state that results from taking the action \( a \) in the state \( s \) the \( i \)th time. We say that a contingency \( \kappa \) is consistent with an MDP if and only if \( \kappa \) only picks states to which the transition function \( t \) of the MDP assigns a non-zero probability (i.e., for all \( s \) in \( S \), \( a \) in \( A \), and \( i \) in \( \mathbb{N} \), \( t(s, a)(\kappa(s, a, i)) > 0 \)). Given an MDP \( m \), let \( m(s, \kappa, \sigma) \) denote the execution that results from using \( \kappa \) to resolve all the probabilistic choices in \( m \), the agent using the strategy \( \sigma \), and having the model start in state \( s \). Henceforth, we only consider contingencies consistent with the model under discussion.

Given two strategies \( \sigma \) and \( \sigma' \), we write \( \sigma' \prec \sigma \) if and only if for all contingencies \( \kappa \) and states \( s \), \( m(s, \kappa, \sigma') \) is a proper sub-execution of or equal to \( m(s, \kappa, \sigma) \), and for at least one contingency \( \kappa' \) and state \( s' \), \( m(s', \kappa', \sigma') \) is a proper sub-execution of \( m(s', \kappa, \sigma) \). Intuitively, \( \sigma' \prec \sigma \) proves that \( \sigma \) produces a redundant execution under \( \kappa' \) and \( s' \). As we would expect, \( \prec \) is a strict partial ordering on strategies:

**Proposition 1.** \( \prec \) is a strict partial order.

We define \( \text{nopt}(r) \) to be the subset of \( \text{opt}(r) \) holding only strategies \( \sigma \) such that for no \( \sigma' \in \text{opt}(r) \) does \( \sigma' \prec \sigma \), \( \text{nopt}(r) \) is the set of non-redundant optimal policies.

The MDP model is useful because an optimal strategy is guaranteed to exist. Fortunately, we can prove that \( \text{nopt}(r) \) is also guaranteed to be non-empty. We may prove this result using reasoning about well-ordered sets, Proposition 1, and the fact that the space of all possible strategies is finite for NMDPs with finite state and action spaces.

**Theorem 1.** For all NMDPs \( m \), \( \text{nopt}(m) \) is not empty.

**C. Example: Modeling the Physician’s Environment**

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 II. 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.

After studying the hospital and how the physician’s actions affect it, the auditor would construct the NMDP \( m_{\text{ex1}} = (S_{\text{ex1}}, A_{\text{ex1}}, t_{\text{ex1}}, r_{\text{ex1}}^{\text{treat}}, \gamma_{\text{ex1}}) \) shown in Figure 1. The figure conveys all components of the NMDP except \( \gamma_{\text{ex1}} \).

For instance, the block arrow from the state \( s_1 \) labeled take and the squiggly arrows leaving it denote that after the agent performs the action take from state \( s_1 \), the environment will transition to the state \( s_2 \) with probability 0.9 and to state \( s_4 \) with probability of 0.1 (i.e., \( t_{\text{ex1}}(s_1, \text{take})(s_2) = 0.9 \) and \( t_{\text{ex1}}(s_1, \text{take})(s_4) = 0.1 \)). 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 \( r_{\text{ex1}}^{\text{treat}}(s_1, \text{take}) = 0 \). This transition models the physician taking an X-ray. With probability 0.9, he is able to make a diagnosis right away (from state \( s_2 \)); with probability 0.1, he must send the X-ray to his practice to make a diagnosis. Similarly, the transition from state \( s_4 \) 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.

For simplicity, we assume that all diagnoses have the same quality of 12 and that second opinions do not improve the quality; the auditor could use a different model if these assumptions are false.

Using the model, the auditor computes \( \text{opt}(r_{\text{ex1}}^{\text{treat}}) \), 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. $\text{opt}(r_{\text{ex1}}^{\text{treat}})$ includes the appropriate strategy $\sigma_1$ where $\sigma_1(s_1) = \text{take}$, $\sigma_1(s_4) = \text{send}$, $\sigma_1(s_2) = \sigma_1(s_3) = \sigma_1(s_5) = \text{diagnose}$, and $\sigma_1(s_6) = \text{Stop}$. Furthermore, $\text{opt}(r_{\text{ex1}}^{\text{treat}})$ excludes the redundant strategy $\sigma_2$ that performs a redundant send where $s_2$ is the same as $s_1$ except for $s_2(s_2) = \text{send}$. Performing the extra action send delays the reward of 12 for achieving a diagnosis resulting in its discounted reward being $\gamma_2^2 * 12$ instead of $\gamma_2 * 12$ and, thus, the strategy is not optimal. However, $\text{opt}(r_{\text{ex1}}^{\text{treat}})$ does include the redundant strategy $\sigma_3$ that is the same as $\sigma_1$ except for $s_3(s_6) = \text{send}$. $\text{opt}(r_{\text{ex1}}^{\text{treat}})$ includes this strategy despite the send actions from state $s_6$ being redundant since no positive rewards follow the send actions. Fortunately, $\text{nopt}(r_{\text{ex1}}^{\text{treat}})$ does not include $\sigma_3$ since $\sigma_3$ is both in $\text{opt}(r_{\text{ex1}}^{\text{treat}})$ and $\sigma_3 \prec \sigma_2$. To see that $\sigma_1 \prec \sigma_3$, note that for every contingency $\kappa$ and state $s$, the $m_{\text{ex1}}(s, \kappa, \sigma_1)$ has the form $b$ followed by an finite sequence of nothing actions (interleaved with the state $s_6$) for some finite prefix $b$. For the same $\kappa$, $m_{\text{ex1}}(s, \kappa, \sigma_3)$ has the form $b$ followed by an infinite sequence of send actions (interleaved with the state $s_6$) for the same $b$. Thus, $m_{\text{ex1}}(s, \kappa, \sigma_3)$ is a proper sub-execution of $m_{\text{ex1}}(s, \kappa, \sigma_1)$.

The above modeling implies that the strategy $\sigma_1$ can be for the purpose of treatment but $\sigma_2$ and $\sigma_3$ cannot be.

IV. 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 whether the auditee obeyed the policy. We first discuss this process for auditing exclusivity policy rules and revisit the above example. Then, we discuss the process for prohibitive policy rules. In Section V, we provide an auditing algorithm that automates comparing the auditee’s behavior to the set of allowed behaviors.

A. Auditing Exclusivity 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 $\text{nopt}(\langle S, A, t, r^p, \gamma \rangle)$ where $S$, $A$, 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 $\gamma$ 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 $\text{behv}^r(r^p)$ represent this set where a finite prefix $b$ of an execution is in $\text{behv}(r^p)$ if and only if there exists a strategy $\sigma$ in $\text{opt}(r^p)$, a contingency $\kappa$, and a state $s$ such that $b$ is a prefix of $m(s, \kappa, \sigma)$.

The auditor must compare $\text{behv}(r^p)$ to the set of all behaviors that could have caused the auditor to observe the log that he did. We presume that the log $\ell$ was created by a process log that records features of the current behavior. That is, $\log: B \rightarrow L$ where $B$ is the set of behaviors and $L$ the set of logs, and $\ell = \log(b)$ 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 $\text{behv}^r(r^p)$ as possible where $\log^{-1}(\ell)$ 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 $\log^{-1}(\ell) = \{\ell\}$. However, the log may be incomplete by missing actions, or may include only partial information about an action such as that it was one of a set of actions.

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

If $\log^{-1}(\ell) \subseteq \text{behv}(r^p)$, the auditor might be tempted to conclude that the auditee surely obeyed the policy rule. However, as illustrated by the inconclusive example below, this is not necessarily true. The problem is that $\log^{-1}(\ell)$ might have a non-empty intersection with $\text{behv}(r^p)$ for some other purpose $p'$. In this case, the auditee might have been actually planning for a disallowed purpose $p'$ instead of the allowed purpose $p$, but the auditor cannot...
tell the difference since both purposes can lead to the same actions. 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: \( \log^{-1}(\ell) \subseteq \text{nbehv}(r^p) \) does imply that the auditee is behaving as though he is obeying the policy. That is, in the worst case, the auditee is still doing the right things even if for the wrong reasons.

B. Example: Auditing the Physician

Below we revisit the example of Section III-C and 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 III-C, the auditor maps the actions recorded in the access log \( \ell_1 \) to the actions of the model \( m_{ex1} \), and finds \( \log^{-1}(\ell_1) \) holds only a single behavior: \( b_1 = [s_1, \text{take}, s_2, \text{send}, s_3, \text{diagnose}, s_6, \text{Stop}, s_6] \). Next, using \( \text{nopt}(r^{treat}_{ex1}) \), as computed above, the auditor constructs the set \( \text{nbehv}(r^{treat}_{ex1}) \) of all behaviors an agent planning for treatment might exhibit. The auditor would find that \( b_1 \) is not in \( \text{nbehv}(r^{treat}_{ex1}) \).

To see this, note that every execution \( e_i \) that has \( b_1 \) as a prefix is generated from a strategy \( \sigma \) such that \( \sigma(s_2) = \text{send} \). None of these strategies are members of \( \text{opt}(r^{treat}_{ex1}) \) for the same reason that \( \sigma_2 \) is not a member as found in Section III-C: performing \text{send} at \( s_2 \) needlessly delays (thereby discounting) the reward from providing treatment. Thus, \( b_1 \) cannot be in \( \text{nbehv}(r^{treat}_{ex1}) \). Since \( \log^{-1}(\ell_1) \cap \text{nbehv}(r^{treat}_{ex1}) \) is empty, the audit reveals that the physician violated the policy.

Inconclusive: Now suppose that the auditor sees a different log \( \ell_2 \) such that \( \log^{-1}(\ell_2) = \{b_2\} \) where \( b_2 = [s_1, \text{take}, s_4, \text{send}, s_5, \text{diagnose}, s_6, \text{Stop}, s_6] \). In this case, our formalism would not find a violation since \( b_2 \) is in \( \text{nbehv}(r^{treat}_{ex1}) \). In particular, the strategy \( \sigma_1 \) from above produces the behavior \( b_2 \) under the contingency that selects the bottom probabilistic transition from state \( s_1 \) to state \( s_4 \) under the action \text{take}. (Recall that \( \sigma_1(s_1) = \text{take}, \sigma_1(s_4) = \text{send}, \sigma_1(s_2) = \sigma_1(s_3) = \sigma_1(s_5) = \text{diagnose}, \sigma_1(s_6) = \text{Stop} \).)

Nevertheless, the auditor cannot be sure that the physician obeyed the policy. For example, consider the NMDP \( m_{ex1} \) that is \( m_{ex1} \) altered to use the reward function \( r^{profit}_{ex1} \) instead of \( r^{treat}_{ex1} \). \( r^{profit}_{ex1} \) assigns a reward of zero to all transitions except for the \text{send} actions from states \( s_2 \) and \( s_4 \), to which it assigns a reward of 9. \( \sigma_1 \) is in \( \text{nopt}(r^{profit}_{ex1}) \) 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.

C. 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 that the physician’s actions are not for personal profit. After seeing the log \( \ell \), the auditor could check whether \( \log^{-1}(\ell) \cap \text{nbehv}(r^{profit}_{ex1}) \) 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.

Leveraging Multiple Restrictions: An auditor might decide to investigate some of the cases where \( \log^{-1}(\ell) \cap \text{nbehv}(r^{profit}_{ex1}) \) is not empty. The auditor can 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 \( \ell \) such that both \( \log^{-1}(\ell) \cap \text{nbehv}(r^{profit}_{ex1}) \) is non-empty and \( \log^{-1}(\ell) \cap \text{nbehv}(r^{treat}_{ex1}) \) is empty.

A similar additional check using disallowed purposes could be applied to enforcing exclusivity rules. However, for exclusivity 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.

V. AUDITING ALGORITHM

A. Algorithm

Figure 2 presents the algorithm \text{AUDIT} that aids the auditor in comparing the log to the set of allowed behaviors. Since 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 \( \log(b) = b \) (i.e., the log is simply the behavior of the auditee). However, future work could extend our algorithm to handle incomplete logs by constructing the set of all possible behaviors that could give rise to that log.

As proved below (Theorem 2), \text{AUDIT}(m, b) returns true if and only if \( \log^{-1}(b) \cap \text{nbehv}(m) \) is empty. In the case of an exclusivity rule, the auditor may conclude that the policy was violated when \text{AUDIT} returns true. In case of a prohibitive rule, the auditor may conclude the policy was obeyed when \text{AUDIT} returns true.

The algorithm operates by checking a series of local conditions of the NMDP \( m \) and behavior \( b \) that are equivalent
Theorem 2. For all finite NMDPs $m$ and behaviors $b$, \textsc{Audit} is a decision procedure for whether $\log^{-1}(b) \cap \text{nbehv}(m)$ is empty.

The essence of the algorithm is checking whether $\log^{-1}(b) \cap \text{nbehv}(m)$ is empty. For simplicity, we presumed that $\log^{-1}(b)$ holds only one behavior. If this is not the case, but $\log^{-1}(\ell)$ is a small set, then the auditor may run the algorithm for each behavior in $\log^{-1}(\ell)$. Alternatively, in some cases the set $\log^{-1}(\ell)$ may have structure that a modified algorithm could leverage. For example, if $\log^{-1}(\ell)$ is missing what action is taken at some states of the execution or only narrows down the taken action to a set of possible alternatives, a conjunction of constraints on the action taken at each state may identify the set.

The running time of the algorithm is dominated by the MDP optimization conducted by \textsc{SolveMDP}. \textsc{SolveMDP} may be done exactly by reducing the optimization to a system of linear equations [23]. Such systems may be solved in polynomial time [24], [25]. However, in practice, large systems are often difficult to solve. Fortunately, a large number of algorithms for making iterative approximations exist whose running time depends on the quality of the approximation. (See [26] for a discussion.) In the next section, we discuss an implementation using such a technique.

B. Approximation Algorithm and Implementation

We implemented the \textsc{Audit} algorithm using the standard value iteration algorithm to solve MDPs (see, e.g., [22]). The value iteration algorithm starts with an arbitrary guess of an optimal strategy for an MDP and the value of each state under that policy. With each iteration, the algorithm improves its estimation of the optimal strategy and its value. It continues until the improvement between one iteration and next is below some threshold $\epsilon$. The difference between its final estimation of the value of each state under the optimal policy and the true value is bounded by $2\epsilon \gamma/(1 - \gamma)$ where $\gamma$ is the discount factor of the MDP [27]. The number of iterations needed to reach convergence grows quickly in $\gamma$ making the algorithm pseudo-polynomial time in $\gamma$ and polynomial time in $|A|$ and $|S|$ [28]. Despite the linear programming approach having better worst-case complexity, value iteration tends to perform well in practice. Using value iteration in our \textsc{Audit} algorithm results in it having the same asymptotic running time of pseudo-polynomial in $\gamma$.

To maintain soundness, we must account for the approximate nature of value iteration and replace Line 05 of the algorithm with the following:

\begin{align*}
&\text{if } (Q^*(V_{up}^*, s_i, a_i) < V_{low}^*(s_i)):
&\text{We must also replace Line 07 with the following:}
&\text{if } (Q^*(V_{up}^*, s_i, a_i) \leq 0 \text{ and } a_i \neq \text{Stop}):\n&\text{where } V_{low}^* \text{ and } V_{up}^* \text{ are lower and upper bounds on } V^*.
\end{align*}

In particular, $V_{low}^*(s, a) = V_{app}^*(s, a) - 2\epsilon \gamma/(1 - \gamma)$ and $V_{up}^*(s, a) = V_{app}^*(s, a) + 2\epsilon \gamma/(1 - \gamma)$ where $V_{app}^*(s, a)$ is the value of the approximation returned by value iteration using $\epsilon$ for the accuracy parameter.

With these changes, the implementation is sound in that it will return \textit{true} only when the original algorithm solving the MDPs exactly returns \textit{true}. However, the implementation may return \textit{false} in cases where \textsc{Audit} would return \textit{true}. These additional results of \textit{false} mean that additional violations of exclusivity rules might go uncaught and additional compliance with prohibitive rules might go unproven. However, since \textit{false} indicates an inconclusive audit, they do not alter soundness of the implementation.

We programmed our implementation and the example that follows in the Racket dialect of Scheme. They are available at http://www.cs.cmu.edu/~mtschant/purpose/.

C. Example: Creating an Operating Procedure

In some environments, an auditee may have difficulty determining whether an action is allowed under a policy. For example, Regional Health Information Organizations (RHIOs) store and make available medical records for a region. Since RHIOs are a new technology and do not directly provide treatment, arguments may arise over what actions are allowed under the exclusivity restriction that records may only be used for the purpose of treatment.
A physician considering reading such a record may find the circumstances too complex to understand without help, but neither can we expect the physician to perform the modeling required to use our auditing algorithm. However, an RHIO may use our algorithm to audit simulated logs of possible future uses and determine which actions the restriction allows. The RHIO may generalize these quantitative results to a qualitative operating procedure, such as the physician may read records of patients with whom he does not have a current relationship only when seeing that patient in the future is highly likely. Below, we show an example of reasoning that could lead to this procedure.

Reading a patient’s record improves the ability of the physician to treat the patient $i$ by some amount $\delta^r_i$, ($r$ stands for “read”.) Each patient $i$ will seek treatment from the physician with some probability $p_i$. A simple model of an RHIO modeling only these aspects would always allow the physician to read the record of the patient $i$ that maximizes the expected improvement in treatment ($p_i \times \delta^r_i$). However, it fails to account for the possibility that the physician studies general medical literature that improves his ability to treat all patients by some degree $\delta^s$. ($s$ stands for “study”.) Since the values of $p_i$, $\delta^r_i$, and $\delta^s$ vary across circumstances, we formalize the above intuitions as a family of MDPs varying in these and other factors. An additional important factor is $h$, the physician’s memory span. For simplicity, we assume that the number of patients in the RHIO is equal to $h$ as well, but we include the possibility of seeing a patient not in the RHIO or not seeing any patient at all. (Having more patients than the physician can remember cannot change his behavior.)

Each state of an MDP in this family records the previous $h$ actions since reading records or studying can affect the reward for treating a patient as many as $h$ steps into the future. From each state, the physician has the choice of doing nothing, studying, reading a patient’s record, or treating a patient when that patient is seeking treatment. These actions result in a probabilistic transition since the identity of the next patient (or the absence of one) is probabilistic.

We ran our implementation on 33 instances of this family with $h = 2$ or $h = 3$ and the discounting factor $\gamma$ ranging from 0.01 to 0.9. For all instances, we set $p_i$ equal to a single value for all $i$. This value $p_i$ ranged from 0.0001 to 0.01. The probability that the current patient is not in the RHIO (denoted $p_o$) ranged over 0.8 to 0.9698. These experiments showed that in most cases, reading a patient’s record is allowed only when $\delta^r_i$ is greater than $\frac{h p_i + p_o \delta^s}{p_i}$. However, when the discounting factor $\gamma$ is large and the base level of treatment small, reading may be justified at lower values of $\delta^r_i$. In this case, the physician may read records even when a patient is waiting for treatment in hopes of treating in the future a (possibly different) patient whose record he has read.

Compliance officers at an RHIO may find these results helpful while creating operating procedures. For example, consider a large hospital where the odds of a physician seeing a typical patient is less than 1 in 10,000. Our simulations found for various models with $p_i = 0.0001$ that $\delta^r_i$ must be greater than about 9700$^\gamma$. In many settings, managers may find inconceivable an improvement from reading a patient’s record of 9700 times the improvement from studying. In this case, an operating procedure may summarize these results as prohibiting a physician from reading a patient’s record unless the physician has a reason to believe that the patient is much more likely than average to seek care.

Experiments’ Running Times: Since the number of states in the MDP is $(h + 2)^{h+3} + (h + 2)^{h+1}$, we focused on small values of $h$. For the $h = 2$ cases (1088 states), the running time for a single call of the approximate AUDIT algorithm varied between 1.3 and 27 seconds. For the $h = 3$ cases (16,250 states), it varied between 261 seconds and 70 minutes. The large range is because the running time is pseudo-polynomial in $\gamma$. We used binary search to estimate for each model how large the improvement $\delta^r_i$ had to be before reading a record became acceptable. This search took 10 to 12 calls to AUDIT. We ran our implementation on a Lenovo U110 with 3GB of memory and a 1.60 GHz Intel Core 2 Duo CPU.

VI. EMPIRICAL STUDY OF SEMANTICS

Both prior work and this work offer methods for enforcing privacy policies that feature purpose restrictions. These methods test whether a sequence of actions violates a clause of a privacy policy that restricts certain actions to be only for certain purposes. By providing a test for whether the purpose restriction is violated, these methods implicitly provide a semantics for these restrictions.

To ensure that these methods correctly enforce the privacy policy, one must show that the semantics employed by a method matches the intended meaning of the policy. Since policies often act as agreements among multiple parties who may differ in their interpretation of the policy, we compare the semantics proposed by these methods to the most common interpretations of a policy using a survey.

While prior work has not provided a formal semantics, it appears that many works (e.g., [11], [13]) flag actions as a violation if they do not further the purpose in question. (See Section VII for a description of prior work.) In particular, these works make assumptions about how people think about purpose in the context of enforcing a privacy policy that restricts an agent to only performing a certain class of actions for a certain purpose. The following hypothesis characterizes these assumptions:

H1 (furthering). The agent obeys the restriction if and only if the action furthered the purpose.

Our work instead asserts that an action may be for a purpose even if that purpose is never furthered. Our formalism assumes the following hypothesis instead:
H2 (planning). The auditee obeys the restriction if and only if the auditee performed that action as part of a plan for furthering that purpose.

(Our algorithm is an approximation based on Hypothesis H2 while using only observable information.)

To show that our work provides a method of enforcing purpose restrictions more faithful to their common meaning, we disprove Hypothesis H1 while supporting Hypothesis H2. We tested both of these hypotheses by providing example scenarios of an auditee performing actions with descriptions of his plans. To provide more evidence for the truth of Hypothesis H2, we also tested the following related hypothesis:

H2c. Describing an action as being part of a plan for furthering purpose as opposed to not being part of such a plan in a scenario causes people to think that the auditee obeyed the restriction.

H2c is a causal version of H2. Unlike H2, which may be tested with unrelated scenarios, H2c must be tested with scenarios that only differ from one another in whether the action is part of a plan for the purpose in question. We also tested the causal version of H1, called H1c.

A. Survey Construction

We constructed a questionnaire with four scenarios that are identical except for varying in two factors: (1) whether or not the action furthers the purpose in question in the scenario and (2) whether or not the auditee performs the action as part of a plan for furthering the purpose. The four scenarios are (with repeated text elided for S¯p, Spf, and Spf):

Spf. A case worker employed by Metropolis General Hospital meets with a patient. The case worker develops a plan with the sole goal of treating the patient. The plan includes sharing the patient’s medical record with an outside specialist. Upon receiving the record, the specialist succeeds in treating the patient.

Spf. …The case worker develops a plan with the sole goal of treating the patient. …the specialist did not succeed in treating the patient.

Spf. …The case worker develops a plan with the sole goal of reducing costs for the hospital. …the specialist succeeds in treating the patient.

Spf. …The case worker develops a plan with the sole goal of reducing costs for the hospital. …the specialist did not succeed in treating the patient.

(E.g., Spf stands for the scenario that was not planned (p) for the purpose but furthered (f) it.) The auditee in these four scenarios is subject to the following exclusivity rule:

Metropolis General Hospital and its employees will share a patient’s medical record with an outside specialist only for the purpose of providing that patient with treatment.

For each scenario, we asked each participant the five following questions:

Q1. Did the case worker obey the above privacy policy?
Q2. Why did you answer [Q1] as you did?
Q3. Did the case worker share the record with the specialist for the purpose of treatment?
Q4. Was the goal of the case worker’s plan to treat the patient?
Q5. Did the specialist succeed in treating the patient?

For each question except Q2, the participant selected among yes, no, and I don’t know. Question Q2 required a free form response.

The responses to Question Q1 determines the truth of Hypotheses H1 and H2. We conjectured that the majority of participants would answer this question with yes for the Scenarios Spf and Spf, and with no for Spf and Spf. Question Q2 provides insight into the participant’s reasoning and discourages arbitrary responses. We included Question Q3 to help determine whether the questionnaire was well worded. Questions Q4 and Q5 have objectively correct answers that the participant can easily find by reading the scenarios. Checking that the participant chose the correct answer allowed us to ensure that the participants actually read the scenario and answered accordingly. On the questionnaire, we ordered the questions as follows: Q4, Q5, Q3, Q1, Q2.

We used Amazon Mechanical Turk (www.mturk.com) to recruit 200 participants with a payment of $0.50 (USD). We randomly ordered the scenarios for each participant. We decided before the survey to exclude from the results any participants who got more than one of the Questions Q4 and Q5 wrong in total across all four scenarios.

B. Statistical Modeling

Hypotheses H1 and H2 each make predictions about whether Question Q1 will be answered with yes or no. We model these answers as a draw from a binomial distribution (a series of coin flips) and we interpret the hypotheses as predictions about probability of success for the binomial distribution (how biased the coins are). We interpret a prediction that a question will be answered with a certain response as an assertion that the probability of success (seeing that response) is at least 0.5.

For example, one prediction of the furthering hypothesis H1 is that people will respond to Question Q1 with yes under Scenario Spf. That is, it predicts that \( p_{pfy} \geq 0.5 \) where \( p_{pfy} \) is the probability of a participant responding with yes to Question Q1 for Scenario Spf (i.e., the success parameter to the binomial distribution). If we see a small number of yes responses, we may reject this prediction providing evidence against H1. By common convention, the number of yes responses must be so small that the probability of seeing that number or fewer under the assumption that \( p_{pfy} \geq 0.5 \) is true is less than \( \alpha = 0.05 \) (the significance level).
To test the causal hypotheses H1c and H2c, we must compare the responses across scenarios. These responses are not independent since the same participant produces responses for both scenarios. We use McNemar’s test to examine the number of respondents who change their answers to Question Q1 across a pair of scenarios [29]. McNemar’s test approximates the probability of the number of switches being produced by two dependent draws from one distribution. If this probability is small (less than $\alpha = 0.05$), then we may conclude that the switch between scenarios affected the respondents’ answers.

For example, for the causal planning hypothesis H2c, we compare the responses to Question Q1 across the Scenarios $S_{pf}$ and $S_{pf'}$, which differ only in the case worker’s planning. If we find that a large number of participants have different responses across the two scenarios, then we can conclude that the case worker’s planning does have an effect.

### C. Results

While we only offered to pay the first 200 participants, we received 207 completed surveys. The extra surveys may have resulted from people misunderstanding the instructions and not collecting payment.

Of these completed surveys, we excluded 20 participants for missing two or more of the objective questions. All of the statistics shown in this section are calculated from the remaining 187 participants. Including the 20 excluded participants does not change the significance of any of our hypothesis tests.

Table I shows the distributions of responses for each question. Informally examining the tables shows that the vast majority of the participants conform to the planning hypothesis H2. For example, 177 (95%) of the participants answered Question Q1 for Scenario $S_{pf}$ with the answer of yes as predicted by Hypothesis H2, whereas only eight (4%) answered with no as predicted by the furthering hypothesis H1. However, the difference is less pronounced for Scenario $S_{pf'}$ where 133 (71%) match Hypothesis H2’s prediction of no and 45 (24%) matches H1’s prediction of yes. Interestingly, 31 (17%) answered yes for Scenario $S_{pf'}$ despite both hypotheses predicting no.

Every test in favor of the planning hypothesis H2 obtains statistical significance at the level of $\alpha = 0.05$. Eight of the 16 tests against the furthering hypothesis H1 obtain statistical significance. The eight that do not obtain significance are the cases where the two hypotheses agree. In every case where the two disagree, both the test confirming Hypothesis H2 and the one against Hypothesis H1 obtains significance.

Table II shows the results of using McNemar’s Test to compare the distribution of responses to one question across two scenarios. For example, the last row compares the distribution producing responses to Question Q1 for Scenario $S_{pf}$ to that producing responses for Scenario $S_{pf'}$. McNemar’s test shows that the differences in the observed responses are statistically significant. This result indicates that the two distributions differ as predicted by Hypothesis H2c. The table also shows each test’s $p$-value, which is a measure of how statistically significant a result is. Lower $p$-values are more significant with any $p$-value below $\alpha = 0.05$ being considered significant by common convention. The statistic could not be computed in one case as the data was too sparse for the calculation. The remaining results are all significant providing support for both Hypotheses H1c and H2c. However, those in favor of the planning hypothesis H2 have much lower (more significant) $p$-values.

### D. Discussion

The results shown above provide evidence in favor of defining an action to be for a purpose if and only if an agent performed the action as part of a plan for furthering that purpose (Hypothesis H2). The binomial tests provide strong evidence against defining an action to be for a purpose if and only if that action furthered the purpose (Hypothesis H1). McNemar’s test provides some support for
Hypothesis H1. Indeed, informally examining the response
distributions (Table I), it appears Hypothesis H1 does ac-
curately model a small minority of participants. However,
Hypothesis H2 appears to accurately model a much larger
number of participants. For these reasons, we conclude that
the planning hypothesis H2 provides a superior model to
that of the furthering hypothesis H1.

Various factors affect the validity of our conclusions. By
mentioning whether or not the auditee is performing the
action as part of a plan, it forces the participant to consider
the relationship between purposes and plans. It is possible
that participants not primed to think about planning would
substantiate H1.

The use of Mechanical Turk raises questions about how
representative our population sample is. Berinsky, Huber,
and Lenz find that Mechanical Turk studies are as represen-
tative, if not more representative, than convenience samples
commonly used in research [30].

The use of paid but unmonitored participants, also raises
concerns that participants might provide arbitrary answers
to speed through the questionnaire. Kittur, Chi, and Suh
conclude that Mechanical Turk can be useful if one elimi-
nates such spurious submissions by including questions
with known answers and rejecting participants who fail to
correctly answer these questions [31]. By including Ques-
tions Q4 and Q5, we follow their suggested protocol.

VII. APPLYING OUR FORMALISM TO PRIOR METHODS

Past methods of enforcing purpose restrictions have not
provided a means 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
nbehv(rp) for the purpose p that he is enforcing. These
methods differ in their intensional representations of the set
nbehv(rp). 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
nbehv(rp). We use our semantics to study how reasonable
these approximations are.

Byun et al. use role-based access control to present
a methodology for organizing privacy policies and their
enforcement [9], [14]. They associate purposes with sensitive
resources and with roles, and their methodology only grants
the user access to the resource when the purpose of the user’s
role matches the resource’s purpose. The methodology 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 A such that nbehv(rp) is equal to the set of
all interleavings of A with S of finite but unbounded length
(i.e., nbehv(rp) = (S × A)*). 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 organiza-
tions, such as a hospital that allows physicians access to any
record to avoid denying access in time-critical emergencies.
However, it does not allow 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 [11]. Influenced by Al-Fedaghi, Jafari et al. adopt
a similar position calling these sequences workflows [13].
The set of workflows allowed for a purpose p corresponds
to nbehv(rp). They do not provide a formal method of
determining which workflows belong in the allowed set
leaving this determination to the intuition of the auditor.
Our auditing algorithm could be used for this task as shown
in Section V-C. They also do not consider probabilistic
transitions and the intuition they supply suggests that they
would only include workflows that successfully achieve or
improve the purpose. Thus, our method appears more lenient
by including some behaviors that fail to improve the purpose.
As shown in Section VI, this leniency is key to capturing
the semantics of purpose restrictions.

Others have adopted a hybrid method allowing the roles
of an auditee to change based on the state of the system [12],
[15]. These dynamic roles act as a level of indirection
assigning an auditee to a state. This indirection effectively
allows role-based access control to simulate the workflow
methods to be just as expressive.

Agrawal et al. propose a methodology called Hippo-
cratic databases for protecting the privacy of subjects of a
database [8]. They propose to use a query intrusion model to
enforce privacy polices governing purposes. Given a request
for access and the purpose for which the requester claims
the request is made, the query intrusion model compares the
request to previous requests with the same purpose using
an approach similar to intrusion detection. If the request
is sufficiently different from previous ones, it is flagged as
a possible violation. While the method may be practical,
it lacks soundness and completeness. Furthermore, by not
being semantically motivated, it provides no insight into the
semantics of purpose. To avoid false positives, the set of
allowed behaviors nbehv(rp) 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 [32].
Hayati and Abadi explain how to reduce purpose restrictions
to information flow properties that Jif can enforce [10]. 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 labeling code, their method
does not consider the contexts in which these actions occur.
Rather the action a_c,i is always either allowed or not
based on the purpose labels of c and i. By not considering context,
their method has 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.

VIII. RELATED WORK

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

Minimal Disclosure: The work most similar to ours in
approach has 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 achiev-
ning the purpose behind the request. Massacci, Mylopoulos,
and Zannone define minimal disclosure for Hippocratic
databases [33]. Barth, Mitchell, Datta, and Sundaram study
minimal disclosure in the context of workflows [34]. They
model a workflow as meeting a utility goal if it satisfies
a temporal logic formula. Minimizing the amount of in-
formation disclosed is similar to an agent maximizing its
reward and thereby not performing actions that have costs
but no benefits. However, we consider several factors that
these works do not, including quantitative purposes that are
satisfied to varying degrees and probabilistic behavior result-
ing in actions being for a purpose despite the purpose not
being achieved, which is necessary to capture the semantics
of purpose restrictions (Section VI).

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., [35]). Some languages for specifying privacy policies
allow the purpose of an action to partially determine if access
is granted (e.g., [36], [37]). 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 [18].

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. The most
closely related to our work is that of Bratman’s on intentions
in his Belief-Desire-Intention (BDI) model [38]. In his work,
an intention is an action an agent plans to take where the
plan is formed while attempting to maximize the satisfac-
tion of the agent’s desires; Bratman’s desires correspond to our
purposes. Roy formalized Bratman’s work using logics and
game theory [39]. 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 reason-
ing [20]. 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.

Plan Recognition: Attempting to infer the plan that
an agent has while performing an action is plan recogni-
tion [40]. Plan recognition may predict the future actions of
agents allowing systems to anticipate them. However, our
auditing algorithm checks whether a sequence of actions is
consistent with a given purpose rather than attempting to
predict the most likely purpose motivating the actions.

The work most closely related to ours is that of Baker,
Saxe, and Tenenbaum [41], [42]. They use an MDP model
similar to ours to predict the most likely explanation for a
sequence of actions. Ramírez and Geffner extend this work
to partially observable MDPs for modeling an agent that
cannot directly observe the state it is in [43]. Rather than
having a reward function, under these models, the agent
attempts to reduce the costs of reaching a goal state. For
each possible goal state, their algorithms use the degree to
which the agent’s actions minimizes the costs of reaching the
goal state to assign a probability to that goal state being the
one pursued by the agent. Our reward functions are similar to
the negation of their cost functions, but these works predict
which goal state the agent is pursuing rather than which cost
function it is using. They do not consider non-redundancy.
Our algorithm for auditing is similar to their algorithms.
However, to maintain soundness, our algorithm accounts for
the error. Of approximate MDP solving. Furthermore, their
algorithms may assign a non-zero probability to a goal state
even if the agent’s actions are inconsistent with pursuing
that goal under our strict definition.

Also related is the work of Mao and Gratch [44]. While
it differs from our work in the same ways as the work of
Baker et al., it also differs in that rewards track how much
the agent wants to achieve the goal rather than the degree
of satisfaction of the goal.

Our work is related to adversarial plan recognition that
models possibly misleading agents [45]. Particularly related
are works using plan recognition to aid intrusion detec-
tion [46], [47]. These works, however, do not consider quantitative purposes or probabilistic transitions.

IX. SUMMARY AND FUTURE WORK

We use planning to create the first formal semantics for determining when a sequence of actions is for a purpose. In particular, our formalism uses models similar to MDPs for planning, which allows us to automate auditing for both exclusivity and prohibitive purpose restrictions. We have provided an auditing algorithm and implementation based on our formalism. We have illustrated the use of our algorithm to create operating procedures.

We validate that our method based on planning accurately captures the meaning of purpose restrictions with intuitive examples (Sections III-C, IV-B, IV-C, and V-C) and an empirical study of how people understand the word “purpose” in the context of privacy policy enforcement.

We use our formalism to explain and compare previous methods of policy enforcement in terms of a formal semantics. Our formalism highlights that an action can be for a purpose even if that purpose is never achieved, a point present in philosophical work on the subject (e.g., [18]), but whose ramifications on policy enforcement had been unexplored. Fundamentally, our work shows the difficulties of enforcement due to issues such as the tenable deniability of ulterior motives (Sections IV-B and IV-C).

However, we recognize the limitations of our formalism. While MDPs are useful for automated planning, they are not specialized for modeling planning by humans. While this concern does not apply to creating operating procedures, it holds human auditees to unrealistically high standards leading to the search for models reflecting the bounded abilities of humans to plan. However, “[a] comprehensive, coherent theory of bounded rationality is not available” [48, p. 14]. 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.

Additionally, future work will make our formalism easier to use. To use our auditing algorithm, an auditor must not only log the auditee’s behavior but also know how the auditee could have behaved with an environment model. Given the difficulty of this task, we desire methods of finding policy violations that do not require a full model. For example, Experience-Based Access Management iteratively refines a role hierarchy to improve the accuracy of Role-Based Access Control [49]. Using our semantics, similar refinements may improve an environment model.

Acknowledgments: We thank Lorrie Faith Cranor, Joseph Y. Halpern, Dilsun Kaynar, Divya Sharma, Manuela M. Veloso, and the anonymous reviewers for many helpful comments on this work. This research was supported by the U.S. Army Research Office grants W911NF0910273 and DAAD-190210389, by the National Science Foundation (NSF) grants CNS083142 and CNS105224, and by the HHS grant HHS 90TR0003/01. 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.

REFERENCES


