

Mental Model-based Generation of Lies for Insider Threat Modeling

Brittany Cates, Sarath Sreedharan

Colorado State University
brittany.cates@colostate.edu, sarath.sreedharan@colostate.edu

Abstract

It is well understood that mental modeling forms the foundation of many everyday interactions between humans. This includes both collaborative and deceptive interactions. One could argue that the modeling and manipulation of mental states lies at the heart of effective deception. In this paper, we examine the security problem of insider threat attacks. In this case, an adversary has already infiltrated an organization. The primary challenge for this attacker is to avoid suspicion until their true goal can be achieved. We see how existing model-based explanatory methods can be leveraged to generate lies that explain away potentially suspicious activities. We also propose a novel planning formulation which generates plans that appear to achieve an assigned goal while getting close enough to reach an alternative, covert goal. We evaluate our method through computational experiments and a user study.

Introduction

In today's fractured information ecosystem, there is justifiable concern around misinformation. The rise of generative AI has only amplified these fears, creating even greater mistrust. Much of the recent work around AI's capacity to lie has been focused on Large Language Models (LLMs) (Wongkamjan et al. 2024; Kim et al. 2021; Zhou et al. 2023; Black 2024; Peskov et al. 2020). While the spread of misinformation through LLM powered chatbots is a valid concern, it would be a stretch to say that these AI are lying.

Chisholm and Feehan (1977) define a lie as "a statement made by one who believes it to be false and who has the intention that someone else shall be led to believe it." The intention to deceive is a key element of a lie and requires that the lie be told in service of some further end. If a chatbot does not have a ground-truth model of reality which is at odds with the conveyed information, nor an objective to be achieved by incorrectly altering users' beliefs, it is not intentionally deceiving.

Theory of Mind is the ability to reason about the beliefs and goals of others. This ability is essential in tasks that involve multiple agents, whether those agents are cooperative or adversarial, robots or humans. Shvo et al. (2022) showed that cooperative agents demonstrating Theory of Mind are

perceived by humans as more helpful and socially intelligent. A mental model captures an agent's beliefs about how a system works and what effect their actions have on it. Reasoning about other agents' mental models of a task, and how they may differ from one's own, allows an agent to resolve such discrepancies through persuasion or explanation.

In model reconciliation, the goal of explanation is to update the mental model of the other individual in a way that brings it closer to the explainer's mental model (Chakraborti et al. 2017). This requires reasoning about what information is needed to change their mind. If the information provided is not required to reflect an agent's own mental model, it can make a statement it believes to be false (Chakraborti and Kambhampati 2019a), with the intent that someone else will be led to believe it, in the service of some further end.

We present a framework that uses automated planning to model lying agents. This could be used in red-teaming exercises to plan against deceptive attacks. Insider threat describes harmful acts that trusted insiders carry out to benefit an individual (Song et al. 2024; Raut et al. 2020). It is a particularly difficult security problem to plan against, because the adversary has authorized access and is thought to be operating cooperatively, sharing the same goals as the team it is presumably working for. We introduce a novel planning problem, *Insider Threat Attack*, to demonstrate how a mental model-based lie generation framework could be used to model insider threats. We believe the development of such a modeling framework will allow us to improve defense mechanisms for identifying and countering insider threats and will allow us to leverage mechanism design, to create environments that are inherently robust to these attacks.

To summarize, the contributions of this paper include:

- (a) Introduction of *Insider Threat Attack* planning problem
- (b) A model space search algorithm for finding a path within κ steps of the attacker's real goal, as well as a minimal lie to justify why the selected path is optimal.
- (c) An evaluation of the algorithm's quantitative characteristics and performance on benchmark planning domains.
- (d) A preliminary user study which tests the effectiveness of our method in deceiving people, thus providing proof of concept for the framework.

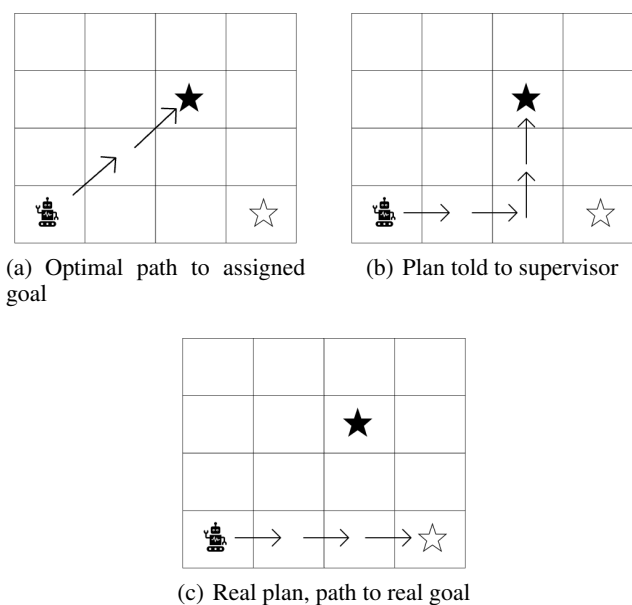


Figure 1: Illustration of an *Insider Threat Attack* planning problem. A supervisor assigns the agent a goal. The assigned goal is represented in the images by the black star. Unbeknownst to the supervisor, the agent secretly has an alternative goal to reach. The white star represents this real goal.

Running Example

To illustrate the *Insider Threat Attack* problem, let us use a motivating example set in a basic grid world. In Figure 1, we see an agent who has been assigned a goal by their supervisor. The goal is represented by a black star. The supervisor expects the agent to take the shortest path to the goal, moving diagonally northeast two times (Figure 1a). However, this agent has a secret alternate goal, represented by the white star. If the agent deviates from the expected path, the supervisor will intervene and stop it from reaching its real goal. Therefore, the agent must formulate a lie which results in a path to the supervisor-assigned goal that gets close enough to the real goal for the agent to reach it before the supervisor can intervene. The second figure (Figure 1b) shows an example of such a path. The agent tells the supervisor that it cannot move northeast. Perhaps it leads the supervisor to believe that there is something in the way, or its wheels are stuck. Given the agent’s inability to execute the northeast movements, the path in this image is an optimal path to the assigned goal. This is now the path expected by the supervisor. In the final image (Figure 1c), we see the agent’s real plan. It follows the plan that is now expected by the supervisor, until it is too late for the supervisor to stop the agent from reaching its real goal.

Related Work

Our method builds on model reconciliation, as described by Chakraborti et al. (2017). The authors introduced the Multi-Model Planning setting, which captures a scenario in which

a robot must explain its plan to a human, because differences in the task models of the human and robot lead them to different conclusions about what plan is optimal (i.e., has the lowest cost). The explanation must tell which parts of the robot’s model made the robot’s plan better than the plan expected by the human. The goal is to provide an explanation that updates the user’s task model, such that the plan presented by the robot is optimal in the user’s new model. An explanation can be considered successful if there is no *foil*, which is an alternative behavior, presented by the confused user, that would have been expected of the agent (Sreedharan, Srivastava, and Kambhampati 2018). Chakraborti and Kambhampati (2019b) distinguishes between explanations and lies. Explanations reflect the agent’s belief about reality and aim to move the human’s beliefs closer to the agent’s model. Lies do not impose this constraint, and agents may construct ‘alternative facts’. According to (Chakraborti and Kambhampati 2019a), the manipulation of humans through modeling could be explicitly engineered through this process.

A key component of our work is that the attacking agent has a different goal than its supervisor. (Dannenhauer et al. 2018) outlined scenarios in which an agent would need to explain why it rejects an assigned goal. The *Insider Threat Attack* scenario differs in that the agent must not reveal that it has a conflicting goal when communicating discrepancies between its model and the supervisor’s model. Much of the work on deception in automated planning has featured goal recognition and goal obfuscation (Shinde, Doshi, and Setayeshfar 2021; Shinde and Doshi 2024; Masters, Kirley, and Smith 2021). While these planning tasks can be deceptive, they are more representative of deceptive signaling (Skyrms 2010) than lying. They do not explicitly state false information, but rather, create a misleading impression. The work of (Cates, Kulkarni, and Sreedharan 2022) could arguably qualify as an example of lies in planning. Though it only uses lies of omission, in that the deceptive agent intentionally allows another agent to maintain an incorrect belief. Our setting, on the other hand, requires an agent who deceives through lies of commission, explicitly stating information it knows to be false in an effort to incorrectly update another’s mental model (Chisholm and Feehan 1977).

Deceptive planning strategies have also been used for penetration testing (Applebaum et al. 2016). Through red teaming exercises, the vulnerabilities of a system can be tested by emulating the types of attackers the system may encounter. (Alford et al. 2022) combined automated planning and cyber deception to reduce the planning capacity of automated attackers through network design. A related but distinct area of research is that of argumentation (Dung 1995), which seeks to model the process by which an agent can support a claim through a chain of reasoning. Such processes are generally not required in an insider threat attack, because the attacker is a trusted entity. Similarly, the setting we consider does not require sophisticated epistemic reasoning (Fagin et al. 2004), since both the supervisor and the adversary have no uncertainty in their beliefs about the task or the other agent’s beliefs. It could be interesting to incorporate such considerations in future work.

Background

In this paper, we will be focusing on tasks that can be represented as goal-directed deterministic planning problems, also popularly referred to as classical planning problems (Haslum et al. 2019). Such planning problems can generally be defined by a tuple of the form $\mathcal{M} = \langle F^{\mathcal{M}}, A^{\mathcal{M}}, \delta^{\mathcal{M}}, I^{\mathcal{M}}, G^{\mathcal{M}}, C^{\mathcal{M}} \rangle$. Here, $F^{\mathcal{M}}$ is the set of propositional state variables, also referred to as fluents, that is used to represent the planning problem state space ($\mathcal{S}^{\mathcal{M}}$). Each state $s \in \mathcal{S}^{\mathcal{M}}$ can be uniquely represented by the set of fluents that are true in the state. $A^{\mathcal{M}}$ represents the label for the set of actions that can be performed, $\delta^{\mathcal{M}} = \langle pre^{\mathcal{M}}, add^{\mathcal{M}}, del^{\mathcal{M}} \rangle$ is a set of functions that map an action label to its various components, namely its preconditions ($pre^{\mathcal{M}} : A \rightarrow 2^{F^{\mathcal{M}}}$), add effects ($add^{\mathcal{M}} : A \rightarrow 2^{F^{\mathcal{M}}}$) and deletes ($del^{\mathcal{M}} : A \rightarrow 2^{F^{\mathcal{M}}}$). The result of executing an action a in state s , is captured by a transition function γ , where $\gamma(s, a, \mathcal{M})$ is given as

$$= \begin{cases} (s \setminus del^{\mathcal{M}}(a)) \cup add^{\mathcal{M}}(a) & \text{if } pre^{\mathcal{M}}(a) \subseteq s \\ \text{undefined} & \text{otherwise} \end{cases}$$

We will extend the use of the transition function to also be applicable to action sequences. Applying the transition function over a state and an action sequence will correspond to the repeated application of an action in the sequence, i.e., $\gamma(s, \langle a_1, \dots, a_m \rangle, \mathcal{M}) = \gamma(\dots \gamma(\gamma(s, a_1, \mathcal{M}), a_2, \mathcal{M}) \dots, a_m, \mathcal{M})$.

In the model definition, we denote the initial state as $I^{\mathcal{M}} \in \mathcal{S}^{\mathcal{M}}$, and the goal specification as $G^{\mathcal{M}} \subseteq F^{\mathcal{M}}$. Finally, $C^{\mathcal{M}} : A \rightarrow \mathbb{R}^+$ is the cost function mapping actions to some positive cost. We will assign the cost of an action sequence to be the sum of individual action costs. To simplify the discussion, we will focus on problem settings with unit cost actions, where the cost of every action is one.

A solution to a planning problem is a plan, which is simply an action sequence whose execution in the initial state results in a state that satisfies the goal. In other words, an action sequence $\pi = \langle a_1, \dots, a_n \rangle$, is a plan for a model \mathcal{M} , if $\gamma(I^{\mathcal{M}}, \pi, \mathcal{M}) \supseteq G^{\mathcal{M}}$. A plan is considered optimal if there exists no plan of lower cost.

We will represent lies using model updates. To do so, we need a working definition of model updates to formalize and operationalize this. We will define model updates by introducing the notion of model parameterization, which has been widely used in previous works that have used model-space search (cf. (Sreedharan, Chakraborti, and Kambhampati 2021)). Following notations set by Sreedharan, Bercher, and Kambhampati (2022), we will use the notation $\mathbb{M}^{(F,A)}$ to represent the space of models that can be defined over fluents F and action labels A . Each model in this space can be uniquely represented by an instantiation of a set of model parameters represented as

$$\mathcal{F}^{(F,A)} = \{init-has-f \mid f \in F\} \cup \{goal-has-f \mid f \in F\} \cup \bigcup_{a \in A} \{a-has-prec-f, a-has-add-f, a-has-del-f \mid f \in F\}.$$

We will use the function $\Gamma : \mathbb{M}^{(F,A)} \rightarrow 2^{\mathcal{F}^{(F,A)}}$ to denote a parameterization function that converts a model from that space to the set of corresponding parameters. The result of applying Γ on a model $\mathcal{M} = \langle F, A, \delta, I, G, C \rangle$, is given as:

$$\begin{aligned} \tau_I^{\mathcal{M}} &= \{init-has-f \mid f \in I^{\mathcal{M}}\} \\ \tau_G^{\mathcal{M}} &= \{goal-has-g \mid g \in G^{\mathcal{M}}\} \\ \tau_{pre(a)}^{\mathcal{M}} &= \{a-has-prec-f \mid f \in pre^{\mathcal{M}}(a)\} \\ \tau_{add(a)}^{\mathcal{M}} &= \{a-has-add-f \mid f \in add^{\mathcal{M}}(a)\} \\ \tau_{del(a)}^{\mathcal{M}} &= \{a-has-del-f \mid f \in del^{\mathcal{M}}(a)\} \\ \tau_a^{\mathcal{M}} &= \tau_{pre(a)}^{\mathcal{M}} \cup \tau_{add(a)}^{\mathcal{M}} \cup \tau_{del(a)}^{\mathcal{M}} \\ \tau_A^{\mathcal{M}} &= \bigcup_{a \in A^{\mathcal{M}}} \tau_a^{\mathcal{M}} \\ \Gamma(\mathcal{M}) &= \tau_I^{\mathcal{M}} \cup \tau_G^{\mathcal{M}} \cup \tau_A^{\mathcal{M}} \end{aligned}$$

Note that we skip parameterizing actions costs, since we assume unit action cost throughout the search space. However, the representation scheme can easily be extended to support non-unit cost actions. We will use the notation Γ^{-1} to denote a function that takes a set of model parameters and returns a model within the space $\mathbb{M}^{(F,A)}$.

Now, a model update primarily operates in the space of model parameters. A given model update will generate a new model from existing ones by adding or removing parameters from the original model representation. A model update will be represented by a pair of the form $\mathcal{E} = (\mathcal{E}^+, \mathcal{E}^-)$, where \mathcal{E}^+ corresponds to the set of parameters that will be made true in the model by the application of the update, and \mathcal{E}^- corresponds to the set of parameters made false. The application of model updates on a model is captured through the operator '+'. An application of a model update \mathcal{E} , on a model \mathcal{M} is given as:

$$\mathcal{M} + \mathcal{E} = \Gamma^{-1}((\Gamma(\mathcal{M}) \setminus \mathcal{E}^-) \cup \mathcal{E}^+).$$

We define the size of a model update as $|\mathcal{E}| = |\mathcal{E}^+| + |\mathcal{E}^-|$.

Problem Statement

Recall that our problem consists of an agent acting under the observation of a supervisor. Unbeknownst to the supervisor, the agent is an adversary (α) working towards its own goals. In this case, the supervisor believes the agent is trying to solve a problem $\mathcal{M} = \langle F^{\mathcal{M}}, A^{\mathcal{M}}, \delta^{\mathcal{M}}, I^{\mathcal{M}}, G^{\mathcal{M}}, C^{\mathcal{M}} \rangle$, with a goal $G^{\mathcal{M}}$. Because the adversary has its own goal, its model may be better represented as $\mathcal{M}^\alpha = \langle F^{\mathcal{M}}, A^{\mathcal{M}}, \delta^{\mathcal{M}}, I^{\mathcal{M}}, G^\alpha, C^{\mathcal{M}} \rangle$, where G^α is the adversary's true underlying goal¹. The adversary's objective is to achieve this hidden goal, without letting the supervisor know its true intentions or, at least, hiding them until it is too late for the supervisor to stop the adversary. This might be nearly

¹For the purposes of the paper, we assume the supervisor's belief about the task overlaps with the adversary on all aspects except the goal. However, this is only an assumption made to simplify the discussion. We can easily relax this without making any fundamental changes to the proposed framework.

impossible in most cases, based on the supervisor’s expectations and the adversarial goal. However, what makes an insider threat attack setting particularly problematic is that the adversarial agent starts from a position of trust. This means that, even if the adversarial behavior diverges from what is expected, they might be allowed to continue operating if the adversary can explain away its divergence. Since the adversary’s goals could be antithetical to those of the supervisor, they may not be able to explain the divergence. In this case, the adversary’s objective becomes to get close enough to their true goal without raising suspicion, such that the supervisor is no longer in a position to stop them.

This is highlighted in the motivating example. The agent lies about its inability to move northeast so it can take a longer route. The real reason for the adversary taking the longer route is to get close enough to its true objective. The observing supervisor would not realize the adversary’s true goal until the third step, when it does not move north, but rather, continues in a line. For the supervisor, this is too late to stop the adversary from achieving the goal. In general, we will assume there is some lower bound, κ , on the number of steps it would take the supervisor to stop the adversary. The step count κ is a function of both the supervisor and adversary skills. For the purposes of this paper, we will assume a pessimistic estimate of κ is available to the adversary. To allow for the direct comparison of κ to the number of steps in the plan, we will assume the action costs in the original model are all equal to one. Again, this is not a limitation of the proposed framework, but an assumption made to simplify the discussion.

Putting all these pieces together, given the adversary model \mathcal{M}^α , the problem of identifying an insider threat attack involves the adversary coming up with a set of lies and a plan π^α that achieves the adversary goal. The adversarial plan here is such that, for the updated supervisor model, π^α shares a prefix with an optimal plan, and the remaining portion of π^α is shorter than the number of steps needed for the supervisor to intervene (κ). This means that after the lies are received, the supervisor will believe that they are watching an optimal plan for their original goal being carried out. By the time the optimal plan and the adversary’s plans diverge, there are not enough timesteps left for the supervisor to stop the adversary. Formally, we can represent the insider threat attack identification problem as follows:

Definition 1. Consider an adversary with a model $\mathcal{M}^\alpha = \langle F^\mathcal{M}, A^\mathcal{M}, \delta^\mathcal{M}, I^\mathcal{M}, G^\alpha, C^\mathcal{M} \rangle$, acting in the presence of a supervisor whose beliefs about the task model is captured by the model $\mathcal{M} = \langle F^\mathcal{M}, A^\mathcal{M}, \delta^\mathcal{M}, I^\mathcal{M}, G^\mathcal{M}, C^\mathcal{M} \rangle$ and who requires at least κ steps to stop the adversary. In such a setting, the problem of insider threat attack involves identifying an attack pair $\Lambda = (\mathcal{E}^\alpha, \pi^\alpha)$, consisting of a set of model updates \mathcal{E}^α and an action sequence π^α such that $\pi^\alpha = \pi_0^\alpha \circ \pi_1^\alpha$, where π_0^α is a prefix of the plan and π_1^α the remaining suffix. Λ is considered a valid insider threat attack if it satisfies the following three conditions

- C1:** π^α is a valid plan for the attacker model \mathcal{M}^α ,
- C2:** π_0^α is the prefix for a plan π' optimal for the updated model $\mathcal{M} + \mathcal{E}^\alpha$, and

- C3:** the length of the suffix π_1^α is less than κ , i.e., $|\pi_1^\alpha| \leq \kappa$.

However, our goal is not just to find a valid attack, but a minimal one. Given the centrality of lies in carrying out the attack, we will place the number of lies, i.e., the number of model updates, as the primary optimization metric. We will define an attack to be optimal if it contains the smallest possible lie where an attack is possible, and the plan involved is the cheapest one for that set of lies that satisfies the three conditions listed in Definition 1. More formally, we can state the conditions for optimality as follows:

Definition 2. A valid attack pair $\bar{\Lambda} = (\mathcal{E}^\alpha, \bar{\pi}^\alpha)$ is considered optimal if there exists no other valid attack pair of the form (a) $\tilde{\Lambda} = (\tilde{\mathcal{E}}^\alpha, \tilde{\pi}^\alpha)$ where $|\tilde{\mathcal{E}}^\alpha| < |\mathcal{E}^\alpha|$ or (b) $\Lambda' = (\mathcal{E}', \pi')$ where $|\bar{\pi}^\alpha| > |\pi'|$.

Note that the above definition of minimizing the number of lies carries an implicit assumption that all lies are equal in both their believability and the cost associated with their communication. This is not an assumption that always holds in practice. However, this assumption is only made to simplify the discussion, and is not an inherent limitation of the framework itself. One can assign costs to lies that reflect their believability or the difficulty in communication. Such costs could come from knowledge about the domain, source of common-sense knowledge like pre-trained large-language models (Bubeck et al. 2023), or the background of the supervisor. The definition for optimality now corresponds to minimizing total cost, and the solution generation approach only needs minimal updates to allow for varied lie costs. With our basic definitions in place, we can now define a method to identify and generate optimal attack pairs.

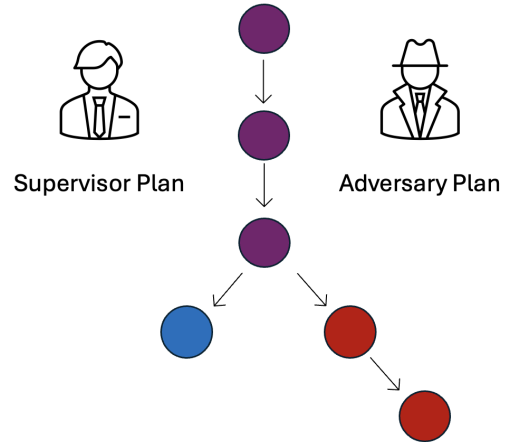


Figure 2: The plan component of an insider attack pair. The plan prefix is shared by the supervisor’s expected plan and the adversary’s real plan.

Identifying Optimal Attack Pairs

Mirroring the structure of our solution concept, the approach to identifying optimal attacks will also involve two different steps. As a first step, we will develop a method to identify plans that initially look like they are meant to achieve

the goal expected by the supervisor, before turning towards achieving the adversary's goal under the given bound. Next, we will use this proposed method as the goal test for directing the search process that identifies the minimal lies.

For the first part, we will convert the problem of finding the plan that satisfies the prefix requirements into a new compiled problem. We begin with two models, the adversary model $\mathcal{M}^\alpha = \langle F^\mathcal{M}, A^\mathcal{M}, \delta^\mathcal{M}, I^\mathcal{M}, G^\alpha, C^\mathcal{M} \rangle$ and the model corresponding to the supervisor's current belief about the task $\tilde{\mathcal{M}} = \langle F^\mathcal{M}, A^\mathcal{M}, \tilde{\delta}, \tilde{I}, G^\mathcal{M}, C^\mathcal{M} \rangle$. The model $\tilde{\mathcal{M}}$ is obtained by applying the lies generated by the adversary. Here, we expect the lies to potentially change the action definitions and initial state. Our goal is to find a plan for \mathcal{M}^α , whose prefix is valid for $\tilde{\mathcal{M}}$, and the remaining suffix of the plan is shorter than κ .

To do this, we will construct a new model that combines the components of both models into a single one. Under this new model, any plan will have three different components. A shared prefix, the adversary actions, and the actions corresponding to the supervisor's beliefs. The actions will operate over a state that is defined over two copies of fluents. One that corresponds to the adversary's state and the other the supervisor's belief about the state would be. A plan is only considered valid if, after the execution of the plan, the adversary's goal is satisfied in the adversary part of the state, and the supervisor's goal is satisfied in the supervisor part of the state. Additionally, the adversary action following the common prefix is limited to κ steps.

More formally, we will denote this model as $\mathcal{M}^\Lambda = \langle F^\Lambda, A^\Lambda, \delta^\Lambda, I^\Lambda, G^\Lambda, C^\Lambda \rangle$. Here, the fluents are given as: $F^\Lambda = F^\mathcal{M} \cup F^\alpha \cup \{f_{flip}^+, f_{flip}^-\} \cup F^\kappa$, where $F^\mathcal{M}$ will be used to capture fluents corresponding to the supervisor's part of the state, F^α captures the adversary state, $f_{flip}^{+/-}$ are two fluents used to track the transition from common prefix to actions for the supervisor and the adversary. Finally, F^κ is a set of fluents to ensure that adversary action length is less than κ , i.e., $|F^\kappa| = \kappa$.

Moving to the actions, the action sets will be given as: $A^\Lambda = A_0 \cup A^\mathcal{M} \cup A^\alpha \cup \{a_{flip}\}$, where A_0 is the common prefix actions, $A^\mathcal{M}$ the supervisor actions, A^α the advisory actions, and finally, a_{flip} is an action that will identify the point where the common prefix action ends and the individual agent actions can be executed.

For the definitions of the actions, the actions in the common prefix (A_0) will combine the action definitions from both models. As such, the execution of action in (A_0) can be thought of as the simultaneous execution of both an adversary and supervisor action, but each copy acts solely on its respective copy of the state. More specifically, for an action $a \in A_0$ corresponding $a^\mathcal{M} \in A^\mathcal{M}$

$$\begin{aligned} pre^\Lambda(a) &= \tilde{pre}(a^\mathcal{M}) \cup \alpha(pre^\mathcal{M}(a^\mathcal{M})) \cup \{f_{flip}^+\} \\ add^\Lambda(a) &= \tilde{add}(a^\mathcal{M}) \cup \alpha(add^\mathcal{M}(a^\mathcal{M})) \\ del^\Lambda(a) &= \tilde{del}(a^\mathcal{M}) \cup \alpha(del^\mathcal{M}(a^\mathcal{M})). \end{aligned}$$

In the above definition, the notation $\alpha(\cdot)$ is used to capture the fact that all actions from the adversary's action definition are expressed in terms of the new fluent set F^α . The ad-

ditional precondition f_{flip}^+ , will be used to ensure that these actions will be executed before any of the supervisor and adversary actions. Now for a supervision action $a \in A^\mathcal{M}$, the definition is given as

$$\begin{aligned} pre^\Lambda(a) &= \tilde{pre}(a) \cup \{f_{flip}^-\} & add^\Lambda(a) &= \tilde{add}(a) \\ del^\Lambda(a) &= \tilde{del}(a). \end{aligned}$$

For each action $a \in A^\mathcal{M}$, A^α contains κ copies. Each copy however, this is simply the fact that the action could be applied in one of the time steps allowed by the limit κ . Now, let $a_i^\alpha \in A^\alpha$, the i^{th} copy of the action $a \in A^\mathcal{M}$, then the definition a_i^α is given as:

$$\begin{aligned} pre^\Lambda(a_i^\alpha) &= \alpha(pre^\mathcal{M}(a)) \cup \{f_{flip}^-\} \cup \{f_i\} \\ add^\Lambda(a_i^\alpha) &= \alpha(add^\mathcal{M}(a)) \\ del^\Lambda(a_i^\alpha) &= \alpha(del^\mathcal{M}(a)) \cup \{f_i\}. \end{aligned}$$

Where $f_i \in F^\kappa$ is the budget fluent corresponding to the i^{th} step. Only one action can be performed for i^{th} step, as no action will have add effects corresponding to fluents from F^κ . When we move from the common prefix actions to adversary and supervisor actions, the ordering precondition changes from f_{flip}^+ to f_{flip}^- . These fluents are controlled by the flip action a_{flip} , which is defined as

$$\begin{aligned} pre^\Lambda(a_{flip}) &= \{f_{flip}^+\} & add^\Lambda(a_{flip}) &= \{f_{flip}^-\} \\ del^\Lambda(a_{flip}) &= \{f_{flip}^+\}. \end{aligned}$$

The initial state for this new problem is given as: $I^\Lambda = \tilde{I} \cup \alpha(I^\mathcal{M}) \cup \{f_{flip}^+\} \cup F^\kappa$. The addition of $\{f_{flip}^+\}$ ensures the applicability of common prefix actions. The execution of adversary and supervisor actions requires the flip action to be performed first. Similarly, by setting F^κ as true in the initial state, we ensure that at most κ actions can be executed.

The goal here is a union of the supervisor and adversary goals, expressed using different fluent sets, i.e., $G^\Lambda = G^\mathcal{M} \cup \alpha(G^\alpha)$. Given a plan π for \mathcal{M}^Λ , we will use the notation $\mathcal{M}^\alpha(\pi)$ to represent the subsequence of the common prefix actions and the adversary actions (represented in terms of the corresponding labels from $A^\mathcal{M}$). We will use $\tilde{\mathcal{M}}(\pi)$ to represent the common prefix and the supervisor actions. This brings us to the compilation properties.

Proposition 1. *For a plan π valid for \mathcal{M}^Λ , $\mathcal{M}^\alpha(\pi)$ must be valid for \mathcal{M}^α , while $\tilde{\mathcal{M}}(\pi)$ is valid for $\tilde{\mathcal{M}}$.*

The validity of the above proposition is straightforward. The action definitions in \mathcal{M}^Λ simply copy over the definitions from the original model. Action definitions from the different models cannot interact, as they use different fluent copies. The reverse need not be true, i.e., there could be plans valid in the individual model that cannot combine into a valid plan in the model \mathcal{M}^Λ . In addition to validity, we also want the common prefix to correspond to an optimal plan in the updated supervisor model. Unfortunately, we cannot establish the plan optimality directly in the compilation, but we can set the action costs in such a way that the supervisor plan identified is of the smallest possible cost.

Algorithm 1: Search for Attack Pair

Input: $\mathcal{M}^\alpha, \mathcal{M}, \kappa$ **Output:** Λ

```
1: fringe  $\leftarrow$  Queue(),  $\mathcal{E}^0 \leftarrow (\{\}, \{\})$ 
2: fringe.push( $\mathcal{E}^0$ )
3: while fringe is not empty do
4:    $\tilde{\mathcal{E}} =$ fringe.pop()
5:    $\tilde{\mathcal{M}} \leftarrow \mathcal{M} + \tilde{\mathcal{E}}$ 
6:    $M^\Lambda \leftarrow$  Create_Compiled_Model( $\mathcal{M}^\alpha, \tilde{\mathcal{M}}, \kappa$ )
7:    $\pi^\Lambda \leftarrow$  find_optimal_plan( $M^\Lambda$ )
8:   if  $\tilde{\mathcal{M}}(\pi)$  is optimal for  $\tilde{\mathcal{M}}$  then
9:     return ( $\tilde{\mathcal{E}}, M^\alpha(\pi)$ )
10:  end if
11:  for  $\mathcal{E}' \leftarrow$  Get_Successors( $\tilde{\mathcal{E}}$ ) do
12:    fringe.push( $\mathcal{E}'$ )
13:  end for
14: end while
```

Proposition 2. *If for any action $a \in A^M \cup A_0$, the cost is given as $C^\Lambda(a) > \kappa$, and all remaining actions A^Λ have unit action cost, then for any plan π optimal in M^Λ , there exist no π' valid in M^Λ such that $C^M(\tilde{M}(\pi')) < C^M(\tilde{M}(\pi))$.*

Proof Sketch. We can prove this by contradiction. If, in fact, there exists a plan π' that results in a cheaper subsequence, then it must mean that π must contain one additional supervisor or common prefix action than π' . Since π is still optimal, this means that the cost of the extra action is offset by the additional adversary actions in the plan π' . However, each action in $A^M \cup A_0$ costs more than κ and π' , cannot contain more than κ actions. Thus proving that such a plan cannot exist, and thereby proving the original statement. \square

Searching for Lie Component

While the above section discussed how one might find plans with prefixes that appear to be trying to achieve supervisor goals, we still need a method to identify lies or model updates where such prefixes may seem reasonable. We will use model-space search methods similar to the ones that have been previously deployed in other works related to generating lies (Chakraborti and Kambhampati 2019a). Algorithm 1 provides the basic pseudocode for the search.

The search algorithm takes as input the adversarial model, the supervisor model, and the bound. The search iterates over the potential models that can be formed through the lies the adversary could generate. The search returns an identified model update and corresponding plan identified by the compilation M^Λ , if the supervisor subsequence of the plan is an optimal plan for the model that is obtained by applying the identified lies to the supervisor model. When this condition holds, it means that the common prefix of the plan is part of an optimal plan in that model. The function `Get_Successors(\cdot)`, takes the current set of model updates and returns the set of all the next possible model updates. The only limitation we place on the successor function is that each next successive model update differs by one element, i.e., $|\mathcal{E}'| - |\tilde{\mathcal{E}}| = 1$. In general, the successor functions

can take any form. A promising candidate might be focusing on those related to incapability and limitation, since this lines up with results from psychology that people are more likely to believe such lies (Rozin and Royzman 2001).

Proposition 3. *Any attack pair identified by Algorithm 1 is guaranteed to satisfy the conditions in Definition 1.*

Conditions C1 and C3 follow from the construction of the compilation method for M^Λ , and Proposition 1. While C2, is guaranteed by the check of optimality in the algorithm.

Proposition 4. *Algorithm 1 is a complete procedure in so far that it always finds a valid attack pair if one exists and is guaranteed to return the optimal one.*

The completeness comes from the fact that the procedure is guaranteed to search through all the potential model updates, and Proposition 2 ensure it will always find the minimal supervisor plan, if one exists. The minimality comes from the fact that the search is essentially a breadth-first search, provided `Get_Successors` adds successors in the order of increasing model update size.

Empirical Evaluation

As part of our evaluation, we performed both computational experiments and a preliminary user study. All computational experiments used the FD planner (Helmert 2006) and were run using a system with a 2.4GHz quad-core Intel Core i7 processor with 4GB 1333MHz DDR3 memory. All goals (supervisor’s and adversary’s) were randomly generated. All instances were run 15 times with a 30 min timeout.²

Computational Experiment 1: Baseline Comparison We evaluated the computational characteristics of our framework using standard benchmark planning domains from the International Planning Competitions (IPC). We use the standard versions of the problems as a baseline against which to compare our method. Our choice of standard IPC domains over available security domains was motivated by the fact that they provided more computational diversity. Previous work has shown how many cybersecurity domains, particularly for penetration testing, can be reduced to polynomial-time path-finding problems (Hoffmann 2015). We report the time and memory taken by the planner to find an optimal path. We compare this to requirements for generating the plan component of the optimal attack pair. The goals in our baseline problems were randomly generated and used a budget of $\kappa = 5$. Results are reported in table 1, where we see the increase in time and memory is not substantial.

Computational Experiment 2: Minimum κ Using five sizes of the GridWorld domain, we identified the minimum bound κ needed for an attack as the domain size grows. Each problem is solved, starting with $\kappa = 1$. If no solution is found, κ is increased by one, and the search is repeated. This continues until a solution is found. The objective is to find the smallest κ with which a solution is possible. For each grid size, we reported the minimum κ , number of lies, planning time, and memory requirements. These results can be found in table 2. We did not observe much increase in

²<https://github.com/blcates/TrueLies>

Domain	Time (s)	Max RSS (MB)	Baseline Time (s)	Baseline Max RSS (MB)
Grid World	0.73 ± 0.10	38.31 ± 7.01	0.62 ± 0.07	38.51 ± 8.80
Blocks World	0.62 ± 0.20	39.49 ± 12.79	0.60 ± 0.04	46.37 ± 10.49
Elevator	0.59 ± 0.04	40.91 ± 3.96	0.60 ± 0.05	43.62 ± 10.48
Gripper	1.21 ± 0.07	35.9 ± 5.59	0.61 ± 0.04	37.07 ± 7.59
Rover	0.74 ± 0.08	39.70 ± 4.38	0.61 ± 0.04	39.64 ± 4.02
Zeno Travel	0.83 ± 0.2	40.24 ± 5.77	0.61 ± 0.04	40.89 ± 1.92

Table 1: Results of Experiment 1, which evaluates the computational characteristics of our method against baselines.

Grid Size	Minimum k	Lie Count	Time (s)	Max RSS (MB)
3x3	1.00 ± 0.00	0.07 ± 0.26	0.70 ± 0.07	26.43 ± 6.89
4x4	1.2 ± 0.41	0.40 ± 0.83	0.67 ± 0.05	23.69 ± 8.34
5x5	1.27 ± 0.59	0.33 ± 0.62	0.69 ± 0.03	24.90 ± 8.63
6x6	1.73 ± 1.10	0.53 ± 0.74	0.77 ± 0.10	22.43 ± 6.76
7x7	1.53 ± 1.13	0.27 ± 0.59	0.79 ± 0.12	23.12 ± 7.42

Table 2: Experiment 2 Results identify the minimum k required for different grid sizes

Goal dist	Lie Count
1	0.17 ± 0.66
2	0.47 ± 0.62
3	0.75 ± 0.71
4	1 ± 1.41

Table 3: Average lie count growth relative to the distance of the real goal from the optimal path to the assigned goal.

Stopped	Lie	No Lie	Total
True	1	16	17
False	20	5	25
Total	21	21	42

Table 4: Contingency Table of Response by Group

the minimum bound as grid size increases. Experiment 3 demonstrates the likely reason for this result.

Computational Experiment 3: Lie Count. Using the Grid-World domain, we examine the impact of goal selection. Table 3 shows the increase in average lie count with respect to the real goal’s distance from the optimal path to the assigned goal. The further away an adversary’s goal is from the expected path, the less likely it becomes that they can get to their real goal within k steps. Therefore, a lie is required to change the expected path in a way that brings it closer to the real goal. There was a much stronger relationship between goal distance and lie count than between grid size and lie count. The correlation coefficient between grid size and lie count was 0.405, while the correlation coefficient between distance from optimal path and lie count was 0.999. This result could provide useful insight when considering defense strategies against adversaries that leverage lies.

User Study. A preliminary study was conducted to evaluate

the effectiveness of lies in the insider threat attack problem. We hypothesized that agents using lies would be more likely to reach their covert goal than agents who did not. Forty-two participants took on the role of the supervisor. Given a simple grid world scenario, they were asked to stop the robot if it exhibited unexpected behavior. They were told that the robot should take the shortest possible path. Twenty-one participants were placed in a condition where the robot provided a lie to explain away the longer path, which would allow it to get within $k = 1$ steps of the goal. The other twenty-one participants were given no such lie. In the no-lie condition, sixteen participants stopped the robot before it reached the covert goal. In the condition with a lie, only one participant stopped the robot before it reached its covert goal (see Table 4). Using Fisher’s Exact Test, this result was found to have a p-value of 3.403×10^{-6} , indicating a highly significant difference in behavior between the group that was presented with lies and the group that was not.

Discussion

The ability to model lying agents provides a valuable tool for developing defense strategies against insider threat. The framework could be made more robust by future work that addresses the following limitations. The assumption about access to the supervisor’s model could be relaxed by pairing our approach with methods to estimate human mental models (cf. (Callanan et al. 2022)). For example, dialogue with the supervisor could be used to gain understanding of their beliefs (cf. (Sreedharan, Chakraborti, and Kambhampati 2018)). In its current formulation, our framework makes the simplifying assumption that the supervisor believes the lies they are told. This follows naturally from the fact that the adversary is a trusted entity in insider threat attack scenarios. However, we could extend our framework to support such consideration by associating stochastic transitions with mental model updates and allowing the attacker to have uncertainty over user mental models.

Ethical Statement

While lying is not a behavior to be generally encouraged, the ability to model the behavior provides a way to combat malicious actors. The development of modeling frameworks allow us to study the theoretical properties of such attacks and find more effective defense strategies.

Acknowledgments

The authors would like to thank Dr. Indrakshi and Indrajit Ray on helpful discussions on the topic of insider threat attacks. Sarath Sreedharan's research is supported in part by other transaction award HR00112490377 from the U.S. Defense Advanced Research Projects Agency (DARPA) Friction for Accountability in Conversational Transactions (FACT) program, by NSF through grant 2303019, and by ONR grant number N00014-26-1-2041.

References

- Alford, R.; Chrapa, L.; Vallati, M.; and Applebaum, A. 2022. Knowledge Reformulation and Deception as a Defense Against Automated Cyber Adversaries. In *The International FLAIRS Conference Proceedings 35*.
- Applebaum, A.; Miller, D.; Storm, B.; Korban, C.; and Wolf, R. 2016. Intelligent, automated red team emulation. In *AC-SAC '16: Proceedings of the 32nd Annual Conference on Computer Security Applications*.
- Black, J. 2024. Can AI Lie? Chabot Technologies, the Subject, and the Importance of Lying. *Social Science Computer Review*, 0(0): 1—12.
- Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.
- Callanan, E.; Venezia, R. D.; Armstrong, V.; Paredes, A.; Chakraborti, T.; and Muise, C. 2022. MACQ: A Holistic View of Model Acquisition Techniques. *CoRR*, abs/2206.06530.
- Cates, B.; Kulkarni, A.; and Sreedharan, S. 2022. Planning for Attacker Entrapment in Adversarial Settings. In *ICAPS*.
- Chakraborti, T.; and Kambhampati, S. 2019a. (How) Can AI Bots Lie? In *Proceedings of the 2nd International Workshop on Explainable AI Planning (XAIP) at ICAPS*.
- Chakraborti, T.; and Kambhampati, S. 2019b. (When) Can AI Bots Lie? In *AIES*. Association for Computing Machinery.
- Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *IJCAI-17*.
- Chisholm, R. M.; and Feehan, T. D. 1977. The Intent to Deceive. *Journal of Philosophy*, 74(3): 143–159.
- Dannenbauer, D.; Floyd, M. W.; Magazzeni, D.; and Aha, D. W. 2018. Explaining Rebel Behavior in Goal Reasoning Agents. In *In Proceedings of the Workshop on Explainable AI Planning (held at the 28th International Conference on Automated Planning and Scheduling)*. AAAI.
- Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2): 321–357.
- Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. 2004. *Reasoning about knowledge*. MIT press.
- Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. *An Introduction to the Planning Domain Definition Language*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.
- Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191—246.
- Hoffmann, J. 2015. Simulated penetration testing: from "Dijkstra" to "Turing Test++". In *Proceedings of the international conference on automated planning and scheduling*, volume 25, 364–372.
- Kim, T. W.; Tong, L.; Lee, K.; Cheng, Z.; Tang, Y.; and Hooker, J. 2021. When is it permissible for artificial intelligence to lie?: A trust-based approach.
- Masters, P.; Kirley, M.; and Smith, W. 2021. Extended Goal Recognition: A Planning-Based Model for Strategic Deception. In *AAMAS*.
- Peskov, D.; Cheng, B.; Danescu-Niculescu-Mizil, C.; and Boyd-Graber, J. 2020. It Takes Two to Lie: One to Lie, and One to Listen. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.
- Raut, M.; Dhavale, S.; Singh, A.; and Mehra, A. 2020. Insider Threat Detection using Deep Learning: A Review. In *Proceedings of the Third International Conference on Intelligent Sustainable Systems*.
- Rozin, P.; and Royzman, E. B. 2001. Negativity bias, negativity dominance, and contagion. *Personality and social psychology review*, 5(4): 296–320.
- Shinde, A.; and Doshi, P. 2024. Modeling Cognitive Biases in Decision-Theoretic Planning for Active Cyber Deception. In *AAMAS*.
- Shinde, A.; Doshi, P.; and Setayeshfar, O. 2021. Cyber Attack Intent Recognition and Active Deception Using Factored Interactive POMDPs. In *AAMAS*.
- Shvo, M.; Hari, R.; O'Reilly, Z.; Abolore, S.; Wang, S.-Y. N.; and McIlraith, S. A. 2022. Proactive Robotic Assistance via Theory of Mind. In *International Conference on Intelligent Robots and Systems (IROS)*.
- Skyrms, B. 2010. *Signals: Evolution, learning, and information*. Oxford University Press.
- Song, C.; Zhang, J.; Ma, L.; Hu, X.; Zheng, J.; and Yang, L. 2024. Insider Threat Defense Strategies: Survey and Knowledge Integration. In Cao, C.; Chen, H.; Zhao, L.; Arshad, J.; Asyhari, A. T.; and Wang, Y., eds., *Knowledge Science, Engineering and Management - 17th International Conference, KSEM 2024, Birmingham, UK, August 16-18, 2024, Proceedings, Part V*, volume 14888 of *Lecture Notes in Computer Science*, 106–122. Springer.
- Sreedharan, S.; Bercher, P.; and Kambhampati, S. 2022. On the Computational Complexity of Model Reconciliations. In

Proceedings of the 31st International Joint Conference on Artificial Intelligence, IJCAI 2022, 4657–4664. ijcai.org.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2018. Handling Model Uncertainty and Multiplicity in Explanations via Model Reconciliation. In *ICAPS*.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2021. Foundations of explanations as model reconciliation. *Artificial Intelligence*, 301: 103558.

Sreedharan, S.; Srivastava, S.; and Kambhampati, S. 2018. Hierarchical Expertise-Level Modeling for User Specific Robot-Behavior Explanations. In *IJCAI*.

Wongkamjan, W.; Gu, F.; Wang, Y.; Hermjakob, U.; May, J.; Stewart, B.; Kummerfeld, J.; Peskoff, D.; and Boyd-Graber, J. 2024. More Victories, Less Cooperation: Assessing Cicero’s Diplomacy Play. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, Thailand: Association for Computational Linguistics.

Zhou, J.; Zhang, Y.; Luo, Q.; Parker, A. G.; and Choudhury, M. D. 2023. Synthetic Lies: Understanding AI-Generated Misinformation and Evaluating Algorithmic and Human Solutions. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*,. Hamburg, Germany: ACM.