Policy Compliance in Collaborative Systems

Max Kanovich
Queen Mary, University of London
mik@dcs.qmul.ac.uk

Paul Rowe, Andre Scedrov
University of Pennsylvania
{*rowep, scedrov*}*@math.upenn.edu*

Abstract

When collaborating agents share sensitive information to achieve a common goal it would be helpful to them to decide whether doing so will lead to an unwanted release of confidential data. These decisions are based on which other agents are involved, what those agents can do in the given context, and the individual confidentiality preferences of each agent. In this paper we consider a model of collaboration in which each agent has an explicit confidentiality policy. We offer three ways to interpret policy compliance (system compliance, plan compliance and weak plan compliance) corresponding to different levels of trust among the agents. We show it is EXPSPACE-complete to determine whether a given system is compliant and whether the agents can collaboratively reach a given common goal. On the other hand, we show it is undecidable to determine whether a given system has either a compliant plan or a weakly compliant plan leading to a common goal. The undecidability results are, in part, a consequence of the flexibility of the model, which allows interpretations of policy compliance that depend on current configurations.

1 Introduction

When organizations or individuals collaborate, they often share a certain amount of information to achieve some desired result. While much of the information may be general and harmless, some of the information is sensitive, such as detailed sales reports at an organizational level or social security numbers and credit card numbers at an individual level. There is an interplay between achieving a collaborative goal on the one hand, and guarding sensitive information on the other. The decision about which pieces of information to share and which pieces to guard is usually based on a number of factors related to trust. Different participants can be trusted with different information in different contexts. Let us consider some examples.

When Alice eats at a restaurant she usually pays with a credit card. In order to pay for the meal she must give her card to the waiter, Bob, who takes it away for validation. She trusts the waiter, Bob, not to write down her card number or share it with anybody else. If she met Bob in a different social situation, however, she might not give him her credit card in that social context. Similarly, she doesn't give her credit card to her friends even though she may trust them. This is because her friends have no legitimate reason to learn her card number. Alice's decision to share her card number is based on necessity to achieve the goal of paying for dinner, and trust in the restaurant process. Both of these factors are dependent on the situation and the individual.

Consider another scenario where two competing companies make a temporary and partial strategic alliance. This might occur, for example, if they want to forecast the direction of the market in a collaborative way. Generally these companies will not trust each other with their business secrets, but they will have to share something to create an accurate forecast. They may first start with a contract that limits how the shared information will be used or who it can be shared with. This serves to change the context of the interaction. They build mutual trust by explicitly limiting their actions to ones which are acceptable by both sides.

In Section 2 we consider a more detailed example focusing on a medical test for a patient. We see how the patient's confidentiality preferences distinguish acceptable flows of information from unacceptable ones.

These examples serve to demonstrate that confidentiality of information is not an absolute notion. This may be viewed as a special case of privacy being dependent on context [4]. Not only does confidentiality depend on the context of an interaction, it also depends on the personal preferences of the agents involved. In these examples the agents have already implicitly or explicitly formulated an idea of which information flows are acceptable and which are unacceptable, or at least undesirable. Thus when attempting to model and analyze these interactions the confidentiality preferences or policies of the agents play a crucial role. The first question to ask about these scenarios is whether the agents are able to reach their common goal in a way that complies with each of the agents' confidentiality policies.

This paper explores an abstract model of collaboration in which we can formally answer that question. In this model, agents share a common goal but have their individ-

ual preferences explicitly written as confidentiality policies. We build on our previous work in this area [21, 22] based on local state transition systems in which locally controlled data are represented by a syntactic partition of predicate symbols. In this model different trust scenarios are represented by different transition systems and confidentiality policies. Each global configuration is comprised of local configurations for each agent together with a public configuration. The agents change the configuration using actions which are constrained to affect only their own local configuration and the public configuration. The agents collaborate to try to reach a configuration which contains some common goal. This framework resembles a multi-agent version of the classical planning problem from the artificial intelligence literature [6, 23, 30, 9, 14]. We borrow some of the terminology and say that a collaborative plan is a sequence of actions leading from the initial configuration to a goal configuration. The use of the term plan emphasizes the fact that agents typically want to find an acceptable sequence of actions before the interaction in order to avoid missteps.

We assume that the agents are further constrained by their own confidentiality concerns. These are expressed explicitly as confidentiality policies, which are essentially sets of configurations that the agents deem undesirable. We call such configurations "critical". We offer three possible interpretations of the policies which may correspond to various levels of trust among the agents. The first two interpretations were proposed in [21], but the last interpretation is new.

The first interpretation of policies is *system compliance*. A system is compliant if there is no way for the agents to reach a critical configuration. This might be appropriate for a situation in which the agents are generally untrusting such as the example above with competing companies. The contract they create can serve to limit each agent's actions in a way that makes any critical configuration unreachable. That is, any sequence of actions is safe to use because no sequence can create a critical configuration.

The second interpretation is *weak plan compliance*. A weakly compliant plan is one which avoids the critical configurations. Weak plan compliance is violated only when the agents actually reach a critical configuration while following a particular plan. This is a much weaker interpretation than system compliance which considers any possible future actions of the agents. This interpretation emphasizes the current knowledge of the agents along a single plan. Weak plan compliance is more appropriate for situations like the one above in which Alice uses a credit card at a restaurant. She trusts the waiter not to write down her card number within the restaurant process even though he can. So although a critical configuration, in which the waiter retains Alice's card number, is reachable, she trusts the waiter to follow the standard restaurant procedure and not use the actions available to him to create the critical configuration.

The final interpretation is simply called *plan compliance*. This interpretation provides an intermediate level of protection between system compliance and weak plan compliance. Intuitively, if a plan is compliant it protects those agents who follow the plan against those who may choose to deviate from the plan. In particular, if any subset of the agents deviate from a compliant plan then as long as the other agents abort, those who deviated can never reach a configuration which is viewed as critical by those who did not deviate. This may also be appropriate for situations involving competing companies. In contrast to system compliance, this interpretation considers only one plan at a time. Instead of implying that any sequence of actions is safe to use, it implies that a specific sequence is safe to use even if everybody else behaves differently.

The confidentiality policies themselves are quite flexible. In particular, they can express notions of current knowledge. For example, Alice's policy can specify that Bob should not be able to learn her *current* password. Her policy allows Bob to learn her old password as long as he does not learn her current password. This is an advantage of a state-based approach to confidentiality as opposed to the more trace-based approach of Non-Interference [35, 17, 31]. Declassification has to be handled very carefully in that setting [32, 36, 33].

The main results of this paper are about the decidability and complexity of determining whether there is a plan leading to a goal which is compliant with all of the agents' confidentiality policies. We determine the complexity of the collaborative planning problem with respect to each of the three interpretations. In our previous work [21, 22] the actions were assumed to be well-balanced. That is, the number of atomic pre-conditions of an action was the same as the number of atomic post-conditions of that action. Intuitively, the size of the global configuration remains constant throughout an interaction. In contrast, in the present work, we assume no such restriction. Since we allow un-balanced actions in this work, the configuration size may grow or shrink dramatically, impacting the complexity of the problems under consideration.

Our results are that it is *EXPSPACE-complete* to determine whether a system is compliant and if so to determine whether it has a plan leading to a goal. On the other hand, we show that it is *undecidable* to determine whether a system has a weakly compliant plan leading to a goal. Similarly it is *undecidable* to determine whether a system has a compliant plan leading to a goal. These are in contrast to the complexity results of [21] in which the first two decision problems (using system compliance and weak plan compliance) were both shown to be PSPACE-complete when the actions are restricted to be well-balanced (see Figure 1). Our results make it clear that these decision problems can be quite sensitive to modeling choices regarding the types of actions available as well as the interpretation of confidentiality policies used. The possible presence of un-balanced actions and the ability of our model to express policies which depend on the en-

tire path taken increase the complexity of the problems under consideration. Specifically, our undecidability results rely, in an essential way, on the fact that the definitions of both plan compliance and weak plan compliance depend on the path through the system the agents follow.

	System Compliance	(Weak) Plan Compliance
Well-balanced Systems	PSPACE	PSPACE
General Systems	EXPSPACE	Undecidable

Figure 1. Summary of Complexity Results.

We also contrast these undecidability results with similar undecidability results from security protocol analysis [13]. In the Dolev-Yao model used to analyze security protocols the attacker is much more powerful than the honest protocol participants. Our current work uses a model in which there is no external attacker, and intuitively, the participants are evenly matched, and they are more capable than the honest participants in security protocols, *e.g.* they may loop and they may have unbounded memory. This "closed-room" setting already provides interesting dynamics which are worth investigating. The Dolev-Yao model allows the participants and the attacker to create nonces, or fresh values, but the model in this paper does not currently allow for such fresh value creation. This is a notable difference because the undecidability results of [13] rely on the use of nonces, whereas we obtain undecidability in this work without them. In some ways, our current model is similar in spirit to some features of the Dolev-Yao model of contract signing protocols [7, 8] if *all* participants are "optimistic."

The rest of the paper is organized as follows. Section 2 reviews the definitions from local state transition systems. In Section 3 we formally present the three interpretations of confidentiality policies. Section 4 shows that system compliance is EXPSPACE-complete. We give the proofs of undecidability of weak plan compliance and plan compliance in Section 5. We discuss related work in Section 6. Finally, we conclude in Section 7 and indicate possible directions for future research.

2 Background

Our point of departure is the formalism of local state transition systems [21, 22], which was designed to express confidentiality policies alongside goals of collaboration. We briefly review the relevant details of that formalism here.

At a high level, the model has four main components. Being an evolving system, there must be a way of describing the configuration of the system at any given moment. Additionally, we describe how the agents transform the configurations from one to another via local actions. In our setting the agents also have some (common) goal of the collaboration, as well as (distinct) confidentiality concerns. These concerns are expressed as confidentiality policies. We now introduce each of these four components one at a time.

Configurations. Building the description of a configuration from the ground up, we have an underlying signature Σ of constants, variables, function symbols, and predicate symbols. We impose a finiteness condition on the signature, namely that it can only express finitely many facts. Thus Σ itself must be finite. Additionally, if we allow function symbols then we must restrict the nesting depth to be finite. Alternatively, we can simply assume Σ has no function symbols. The results of this paper are unaffected by this choice.

Definition 1 *A* fact *is a ground, atomic predicate over multi-sorted terms.*

Facts have the form, $P(\bar{t})$ where $\bar{t}$ is a tuple of terms containing no variables. A configuration is a multiset of facts. We take both XY and X, Y to denote the multiset union of multisets X and Y. For any fact P, we use P^k to denote $\underbrace{P, P, \ldots, P}_{\text{n times}}$ and P^0 to indicate there are no instances of the fact P.

The global configuration is divided into different local configurations each of which is *accessible* to a specified agent. There is also a public configuration that is publicly accessible to all agents involved in the collaboration. We are considering interactions that take place in a "closed-room" setting, so we can ignore concerns about an outside intruder. The separation of the global configuration is done via a partition of the predicate symbols. We typically annotate local predicate symbols with the identity of the agent who owns it. So, for example, Alice's local facts will look like $P_A(\bar{t})$ or even perhaps just $A(\bar{t})$. The interpretation is that Alice has the information $\bar{t}$ kept locally. Any predicate symbol not explicitly annotated with an identity is public. In this way, the predicate symbols act like fields of a database in which information may be stored.

The global configuration is the multiset union of the local configurations and the public configuration:

$$Z = Z_{A_1}, Z_{A_2}, \ldots, Z_{A_n}, Z_{pub}$$

where Z_{A_i} is the local configuration of agent A_i and Z_{pub} is the public configuration. More generally, we use X_A to denote any multiset of facts all of which are local to agent A, and X_{pub} to denote any multiset of facts all of which are publicly accessible.

Local Actions. Each agent has a finite set of actions which transform the global configuration. Under the interpretation of the local predicates as accessibility restrictions it is natural

to restrict the actions to conform to that interpretation. Thus the actions are local in the sense that each action can only depend on the local facts of at most one agent. They must have the form

$$X_A X_{pub} \rightarrow_A Y_A Y_{pub}$$

where the agent who owns the action (indicated by the subscript on the arrow) is the same agent who owns any local facts mentioned (on both the right and left of the arrow). Another way of viewing this restriction is that agent A does not have either read or write access to any of the predicates which are local to other agents.

The actions work like re-writing rules. $X_A X_{pub}$ are the pre-conditions of the action and $Y_A Y_{pub}$ are the post-conditions of the action. The pre-conditions must be present in the configuration for the action to be *enabled*. By applying the action, the pre-conditions are erased and replaced with the post-conditions. The rest of the configuration remains untouched. Thus, we can apply the action $X_A X_{pub} \rightarrow_A Y_A Y_{pub}$ to the global configuration $V X_A X_{pub}$ to get the global configuration $V Y_A Y_{pub}$.

Definition 2 *A* local state transition system $T = (\Sigma, I, R)$ *is a triple where* Σ *is the finite signature of the underlying language,* $I = \{A_1, \ldots, A_n\}$ *is a finite set of agents and* R *is a set of local actions owned by those agents.*

In general, these may be first order actions in which all free variables are implicitly universally quantified. For the purposes of this paper we only need to consider systems with a finite set of propositional actions, that is, actions without any variables.

In [21], actions were assumed to be well-balanced. That is, the number of facts in the pre-condition of an action was the same as the number of facts in the post-condition of the action. The reason for placing such a restriction is to minimize the complexity of the system without sacrificing too much expressivity. In this paper, however, we allow general actions which might be un-balanced. For example an agent might have an action that copies a fact: $P(t) \rightarrow_A P(t)P(t)$. Similarly, an agent may have an action which erases a fact: $P(t)P(t) \rightarrow_A P(t)$. Both of these actions are un-balanced and they grow and shrink the configuration size respectively. The results of this paper underscore the impact of the choice between systems with only well-balanced actions and systems which allow un-balanced actions.

There is a small detail that deserves some attention. In the standard Dolev-Yao model, the intruder is always assumed to have the same set of basic actions which includes reading any message from the network. This is a useful assumption because it considers the worst-case scenario. In our setting, we do not force every agent to have such actions, although we certainly do not preclude such a situation. Thus, if a term appears in a public predicate, it may not be accessible to every agent. The idea is to have a formalism that is as general as possible by limiting the number of assumptions we have about it. Although we, as modelers, do not assume the agents have all the Dolev-Yao actions, the agents themselves might assume the others *do* have those actions. This assumption could be reflected their decision about which configurations are considered to violate their confidentiality as described later in this section.

Goals of Collaboration. Since we are considering situations in which the agents are collaborating, we need a way of expressing the goals of collaboration as well as the reachability of those goals.

Let us start with some notation to represent exact reachability. We let $X \rhd_T^n Y$ indicate that the configuration Y is exactly reachable from the configuration X by applying n actions from the system T. That is, using actions from T the system can transform from X into Y without any extra facts. $X \rhd_T^* Y$ means Y is exactly reachable from X by applying 0 or more actions. We will later need to talk about configurations which are reachable using the actions from all agents except for one. Thus we write $X \rhd_{-A_i}^* Y$ to indicate that Y can be reached exactly from X without using the actions of agent A_i, (the system T is implied). We also write $X \rhd_t^1 Y$ to mean that an action labeled by t performs the transformation in one step. We may drop the subscript T if the system is clear from the context.

Typically the agents will not care about the entire description of a goal configuration. It will usually be sufficient for a configuration to contain certain facts. For example, in contract negotiations the text of the contract is not known at the outset, but the goal is to have a signed contract. For this reason the notion of a partial goal is better suited for this setting. For this purpose we use the notation $X \rightsquigarrow_T^* Y$ to indicate that a configuration *containing* Y is reachable from X by applying 0 or more actions from the system T. That is, there is some multiset of facts U such that $X \rhd_T^* YU$. Similarly, we write $X \rightsquigarrow_{-A_i}^* Y$ to indicate that this partial goal is reachable without using the actions of A_i, and $X \rightsquigarrow_t^1 Y$ to indicate that action t performs the transformation in one step. In this paper, all goals are partial goals, and reachability is interpreted in this weaker sense unless otherwise stated.

Definition 3 *A* plan *based on* T *is a sequence of configurations* $X_1, X_2, \ldots, X_n$ *such that* $X_i \rhd_T^1 X_{i+1}$ *for* $1 \leq i < n$. *The configurations* X_i *are* contained *in the plan.*

Thus, if a system satisfies $X \rightsquigarrow_T^* Y$, then the system has a plan leading from X to Y. We also say that Y is reachable from X. The use of the word plan is to indicate that it is a suggested course of action. Given a plan, each of the agents may choose to follow the plan or deviate from the plan. Although the words "plan" and "execution" mean the same at a technical level, we use the former to emphasize the difference between the suggested course of action and the course of action which is actually taken.

What does it mean for a configuration to be a goal configuration? We take a flexible approach to this issue. It seems reasonable to assume that collectively the agents have a way of looking at any configuration and deciding if it meets the goal requirements or not, that is, if it contains a goal configuration or not. At the technical level, we assume that the set of goal configurations is decidable. We put no *a priori* limitations on the complexity of the procedure which decides the set of goals, although one can typically assume that it is efficient (polynomial time). For the purposes of this paper, however, it is sufficient to consider the more restrictive case of a finite set of goal configurations that the agents agree upon ahead of time.

Confidentiality Policies. Throughout a typical collaboration, some information must be revealed and other information should be guarded. Also, the information which is shared is often only shared selectively with the agents who need it to successfully achieve the goals of collaboration.

For example, consider a patient who needs to have a medical test performed at a hospital. The process begins with the patient registering with the receptionist who anonymizes the patient with some ID number. A nurse then gets a test sample from the anonymized patient (*e.g.* draws blood, obtains DNA, etc.) to send off to the lab. A lab technician performs the test and sends the results to the patient's physician who then gives the appropriate diagnosis and/or prescription to the patient. The patient has some preferences regarding which agents learn (or can learn) certain combinations of data. For example, the lab technician must learn the result of the test and be able to connect it to his ID number. But the patient does not want the technician to connect the test result to his true identity. The patient will have similar but distinct preferences regarding the other agents involved in the process. These preferences are expressed as partial configurations that the patient wants the system to avoid.

We therefore allow each agent A_i to specify the set of partial configurations which he or she deems to be undesirable or critical.

Definition 4 *The* confidentiality policy *of A_i, or simply the* policy *of A_i is the set of configurations that A_i specifies as* critical.

Thus, in the above example, if the lab technician is identified by LT then the patient's policy will contain for example $P_{LT}(test\_result, patient\_name)$.

Definition 5 *If a global configuration C contains a partial configuration from A_i's policy then C is said to be* critical for A_i. *C is simply called* critical *if it is critical for some agent.*

The set of critical configurations is thus the union over all agents A_i of the set of configurations critical for A_i. These policies should be set in accordance with the existing laws.

Notice that these policies can be very expressive. For example, Alice does not want Bob to know her *current* password. If Alice's password is '15' at one point and then she changes it to '21', then she doesn't care if Bob eventually learns that her password *was* '15'. That is, she views both the configuration $P_A(A, 15, Pwd), P_B(A, 15, Pwd)$ and the configuration $P_A(A, 21, Pwd), P_B(A, 21, Pwd)$ to be critical, but she will accept either the configuration $P_A(A, 15, Pwd), P_B(A, 21, Pwd)$ or the configuration $P_A(A, 21, Pwd), P_B(A, 15, Pwd)$.

We take the same approach to expressing confidentiality policies as we did for expressing goal configurations. Namely, we simply assume each agent A_i has a procedure which can look at a global configuration and determine if it is critical for A_i. That is, the policies are decidable. Again, we place no *a priori* bound on the complexity of the procedure, but one can typically assume the procedure works in polynomial time. Just as with goal configurations, it is sufficient for this paper to consider the more restrictive case of policies which are finite sets of critical configurations.

3 Interpreting Confidentiality Policies

The agents' confidentiality policies are simply sets of configurations which the agents want to avoid. However, they may want to avoid them in a looser or stricter sense depending on the level of trust the agents have amongst themselves. We therefore propose several ways to interpret these confidentiality policies. The first two interpretations were introduced in [21] while the third interpretation is new. In this section we describe each of these interpretations and state the corresponding computational problems.

Definition 6 (System Compliance) *A system in initial configuration W is called* compliant *if the set of reachable configurations from W contains no configuration which is critical for any agent.*

Compliant systems may be viewed as well-designed systems. Each agent has a guarantee that the actions of the system do not allow their critical configurations to be reached, whether by the malicious collusion of other agents or by the careless use of their own actions. A compliant system dispenses with all confidentiality concerns before the collaboration even starts, and the agents can then focus on the separate problem of finding a plan which leads to a goal configuration. Let us consider the simplest example.

Suppose agent A has a term t which should never appear in the public setting. Then A's policy will indicate that any configuration containing $N(t)$ is critical since the predicate $N()$ is publicly accessible. However, A might have the action $P_A(x) \rightarrow_A N(x)$ which publishes any value from the predicate $P_A()$. In this case, the system is definitely not compliant when starting in an initial configuration which contains $P_A(t)$. Since A has the ability to publish t, she can

single-handedly create a configuration which is critical for her.

This interpretation is most appropriate when the agents share a very low level of trust. For example, a company may ask employees or external collaborators to sign a non-disclosure agreement. This effectively deletes a whole class of actions from the system, severely restricting the reachability space. However, for many scenarios this interpretation of policies might be too strong. For example, Alice may want to include the action $P_A(x) \rightarrow_A N(x)$ in order to publish other pieces of information. For this reason it is useful to consider weaker interpretations that can still provide some level of protection. We consider two other notions: plan compliance and weak plan compliance. We first introduce the weak version to motivate the stronger version.

Definition 7 (Weak Plan Compliance) *A plan is said to be* weakly compliant *if none of the configurations contained in the plan are critical for any agent.*

This interpretation may be appropriate for a set of agents who are mutually trusting. If a system has a compliant plan leading to a goal then each agent knows two things. First, they know that there is plan leading from the initial state to the goal. Secondly, they know that none of their critical configurations will be reached *as long as everybody follows the plan.*

This definition emphasizes the *current* configuration of the system. A policy is violated only when a critical configuration is actually reached. This is a big difference from system compliance in which policies are violated as long as a critical configuration is reachable. It is not too difficult to see that a system is compliant if and only if every plan in the system is weakly compliant.

A natural situation where weak plan compliance might be appropriate is when Alice uses a credit card to pay for a meal at a restaurant. She certainly does not want the waiter to write down her credit card number to use later, but there is no mechanism in place to prevent the waiter from doing so. In this scenario Alice can pay for her meal without the waiter learning her card number, but she must trust him not to write it down. A proper formalization of this system is not compliant according to Definition 6, but the system does have a weakly compliant plan.

In order for the notion of weak plan compliance to be appropriate the agents need to have a level of trust that will not be present in more sensitive settings, so interpreting the policies via weak plan compliance may be too weak. For this reason we introduce our last possible interpretation for policies which provides an intermediate level of protection between system compliance and weak plan compliance.

Definition 8 (Plan Compliance) *A plan is said to be* compliant *if it is weakly compliant and if for each agent A_i, and for each configuration Z contained in the plan, whenever $Z \rhd^*_{-A_i} V$, then V is not critical for A_i.*

A compliant plan gives each agent a stronger guarantee than a weakly compliant plan. Let us consider this guarantee from the perspective of the single agent A. First, she knows that the plan itself does not contain any configurations critical for her because it is also weakly compliant. Secondly, she knows that starting from any configuration Z in the plan, all configurations which are reachable from Z using only the other agents' actions will also not be critical for her. These guarantees hold for each agent.

Thus each agent knows that *as long she follows the plan* the other agents cannot collude to create a configuration critical for her. The agents no longer have to trust one another. As soon as one agent deviates from the plan, the other agents may choose to stop their participation. They can do so with the knowledge that the remaining agents will never create a configuration critical for those agents that aborted.

Such a plan has the flavor of a Nash equilibrium in game theory. No agent has any incentive to deviate from the given plan. If agent A does deviate, she only introduces the possibility that the other agents can now collude to create a configuration critical for her, because the guarantee of compliance only exists for the specified plan. Exploring the connection between Nash equilibria and this current interpretation of policies is an interesting avenue of investigation, however, we leave it for future research as it will likely involve some modifications of our formalism.

The main results of this paper are about the decidability and complexity of determining if a goal configuration is reachable in a way which is compliant with all the agents' policies. There are three computational problems for which we would like to determine the decidability and complexity, corresponding to the three interpretations of these policies.

Problem 1: System Compliance. Given a finite set of propositional actions, a finite set of goal configurations, a finite set of critical configurations, and an initial configuration W, determine if the system is compliant, and if so, determine if there is a plan leading from W to one of the goals.

Problem 2: Weak Plan Compliance. Given a finite set of propositional actions, a finite set of goal configurations, a finite set of critical configurations, and an initial configuration W, determine if the system has a weakly compliant plan leading from W to one of the goals.

Problem 3: Plan Compliance. Given a finite set of propositional actions, a finite set of goal configurations, a finite set of critical configurations, and an initial configuration W, determine if the system has a compliant plan leading from W to one of the goals.

4 System Compliance

In this section we demonstrate that Problem 1 (system compliance) is EXPSPACE-complete. The main step on the way to proving this is to show that the coverability problem for Petri nets is equivalent to the reachability problem for local state transition systems (LSTSs). In order to prove this we rely on reductions both to and from the coverability problem for Petri nets, which was shown to be EXPSPACE-hard by Lipton [29], and shown to have an exponential space algorithm by Rackoff [34]. Both of these results are actually about vector addition systems [26], but vector addition systems have been shown to be computationally equivalent to Petri nets [20, 40]. Because of the well-known correspondence between Petri nets and certain fragments of linear logic [15, 24], as well as the connection between LSTSs and linear (affine) logic [21, 16], we only indicate how the correspondence works at a high level. For completeness we include the details of the correspondence in Appendix A. However, to indicate the prominent connections we must first review some definitions from Petri nets. A number of equivalent definitions exist but we use the ones found in [40].

Petri Nets. A *Petri net* is a tuple $N = (P, T, \phi)$ where P is a finite set of *places*, $P = \{p_1, \ldots, p_k\}$, T is a finite set of transitions, $T = \{t_1, \ldots, t_s\}$, and ϕ is a flow function $\phi : (P \times T) \cup (T \times P) \to \mathbb{N}$. A *marking* μ of the Petri net is a function $\mu : P \to \mathbb{N}$ which assigns some number of *tokens* to each place. For two markings μ, μ' we write $\mu \geq \mu'$ whenever $\mu(p) \geq \mu'(p)$ for all $p \in P$.

A transition t is *enabled* at μ if and only if for all $p \in P$, $\phi(p,t) \leq \mu(p)$. When t is enabled then it may *fire* by removing $\phi(p,t)$ tokens from each place p and by adding $\phi(t,p)$ tokens to each place p. We then write $\mu \xrightarrow{t} \mu'$ where $\mu'(p) = \mu(p) - \phi(p,t) + \phi(t,p)$ for all $p \in P$. (This notation also implies that t is enabled at μ.) Thus the set $\{\phi(p,t) \mid p \in P\}$ act like pre-conditions for the transition t and the set $\{\phi(t,p) \mid p \in P\}$ act like post-conditions for the transition t. A firing sequence $\sigma = t_1 \ldots t_n$ is enabled at μ_0 if and only if $\mu_0 \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \mu_n$, for some markings $\mu_1, \ldots, \mu_n$. In that case we write $\mu_0 \xrightarrow{\sigma} \mu_n$.

A marking μ is *reachable* from μ_0 if $\exists \sigma \in T^*$ such that $\mu_0 \xrightarrow{\sigma} \mu$. The reachability set of a marking μ_0 is $R(N, \mu_0) = \{\mu \mid \exists \sigma \in T^*, \mu_0 \xrightarrow{\sigma} \mu\}$. The marking μ is *coverable* from μ_0 if $\exists \mu' \in R(N, \mu_0)$ such that $\mu' \geq \mu$. The coverability set of μ_0 is $C(N, \mu_0) = \{\mu \mid \exists \mu' \in R(N, \mu_0), \mu' \geq \mu\}$. The coverability problem for Petri nets is the following. Given a Petri net N and two markings μ_0, μ_1, decide if $\mu_1 \in C(N, \mu_0)$.

It is possible to show that the coverability problem for Petri nets is computationally equivalent to the reachability for LSTSs. This involves reductions both ways between Petri nets and LSTSs. The details are given in Appendix A, but we give a high level description of the correspondence. Roughly, places of the Petri net correspond to facts of the LSTS (of which we assume there are only finitely many due to the finiteness of Σ). Petri net markings μ correspond to LSTS configurations $\hat{\mu}$. Petri net transitions t_i correspond to LSTS actions $\hat{t}_i$ in such a way that if markings μ and μ' correspond to configurations $\hat{\mu}$ and $\hat{\mu}'$ respectively, then $\mu \xrightarrow{t_i} \mu'$ if and only if $\hat{\mu} \rhd^1_{\hat{t}_i} \hat{\mu}'$.

The reason we use the coverability problem is because we LSTS reachability refers to partial configurations. This allows us to ignore part of a configuration in a way that the coverability let us ignore extra tokens in a marking. This is important because if the correspondence had been with the Petri net reachability problem, the upper bound would be a lot higher; the best upper bound known for Petri net reachability is primitive recursive in the Ackermann function.

Using the two translations from Appendix A we get the following.

Proposition 1 *Let μ_0 and μ_1 be markings of a given Petri net N, and let $\hat{\mu}_0$ and $\hat{\mu}_1$ be the corresponding configurations of the LSTS T_N constructed from N. Then $\mu_1 \in C(N, \mu_0)$ if and only if $\hat{\mu}_0 \rightsquigarrow^*_{T_N} \hat{\mu}_1$.*

Proposition 2 *Let μ_0 and μ_1 be configurations of a LSTS T, and let $\hat{\mu}_0$ and $\hat{\mu}_1$ be the corresponding markings of the Petri net N_T constructed from T. Then $\mu_0 \rightsquigarrow^*_T \mu_1$ if and only if $\hat{\mu}_1 \in C(N_T, \hat{\mu}_0)$.*

Because the reductions are efficient, we conclude that the LSTS reachability problem has the same complexity as the Petri net coverability problem. Lipton's exponential space lower bound [29] and Rackoff's exponential space upper bound [34] combine to imply that the reachability problem for LSTSs is EXPSPACE-complete. We use the complexity of the LSTS reachability problem in order to determine the complexity of Problem 1 (system compliance).

Theorem 1 (System Compliance) *Let T be a local state transition system with a finite set of propositional actions, $\mathcal{Z}$ a finite set of goal configurations, $\mathcal{C}$ a finite set of critical configurations, and W an initial configuration. It is EXPSPACE-complete to determine if the system is compliant, and if so to determine if there is a plan leading from W to one of the goal configurations.*

Proof. We already achieve EXPSPACE-hardness in the case where $\mathcal{C}$ is empty and $\mathcal{Z}$ has a single element Z. This is simply a direct application of the EXPSPACE-hardness of goal reachability in local state transition systems.

In order to show EXPSPACE membership we rely on the fact that EXPSPACE = COEXPSPACE ([38] chapter 8). This means that given a single critical configuration C, it is possible to determine in exponential space whether or not C is

unreachable, that is, whether or not the system is compliant with respect to C. In order to determine system compliance with respect to a finite set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of critical configurations, we can simply create a process which checks compliance with each of the configurations in order. If this process ever determines that the system is not compliant with respect to some C_i then it outputs "no". If it determines that the system is compliant with respect to all of the configurations, then it proceeds to determine if one of the goals is reachable.

Given the set $\mathcal{Z}$ of goal configurations, another process can determine, in order, whether each configuration is reachable. If the process finds a goal which is reachable, it outputs "yes". If all the goals are unreachable, it outputs "no". Since the checks for (non-)reachability are done sequentially, and each one is performed within exponential space, the whole process can be performed in exponential space. ■

Compare Theorem 1 with the PSPACE result of [21] for systems with well-balanced actions (see Figure 1). Lifting the restriction and allowing un-balanced actions causes the problem to jump from PSPACE to EXPSPACE. Also note that using the techniques from [21] we can schedule a plan if one exists without increasing the complexity.

5 Undecidability

In this section we show that General Weak Plan Compliance and General Plan Compliance are both undecidable. The proof is by reduction from two-counter Minsky machines. The reduction we use is modeled after a similar reduction by Kanovich [25] to prove the undecidability of pure monadic linear logic. There are several key differences however. First, Kanovich's proof is about first-order linear logic, and hence takes advantage of the existential quantifier to "create new values." We have no such operator, thus our proof corresponds more closely to a propositional setting. In particular, this is *not* a reduction from Kanovich's result, but rather a direct reduction from Minsky machines.

Second, we point out that since our notion of reachability refers to *partial* goals, reachability more closely corresponds to derivability of certain sequents in affine logic as described in [21]. While full propositional linear logic is known to be undecidable [28], it has been shown that full propositional affine logic is decidable [27]. Thus, although our setting relates to propositional affine logic, the satisfaction of both plan compliance and weak plan compliance differs enough from simple reachability to affect the decidability of the problem.

5.1 Minsky Machines

We use a standard two-counter Minsky machine M of a certain form. We assume the instructions alternate between instructions for register 1 and instructions for register 2. Instructions labeled by a_i will be 'run' by Alice, instructions labeled by b_j will be 'run' by Bob.

Jump	a_i: **goto** b_j;
Add	a_i: $r_1 := r_1 + 1$; **goto** b_j;
Subtract	a_i: $r_1 := r_1 - 1$; **goto** b_j;
0-test	a_i: **if** $(r_1 = 0)$ **goto** b_j **else goto** b_k;
Jump	b_j: **goto** a_k;
Add	b_j: $r_2 := r_2 + 1$; **goto** a_k;
Subtract	b_j: $r_2 := r_2 - 1$; **goto** a_k;
0-test	b_j: **if** $(r_2 = 0)$ **goto** a_k **else goto** a_l;

No two instructions are labeled with the same label. States a_1 and a_0 are the *initial* and *final* states of M, respectively. Furthermore, a_0 is a halting state so it is distinct from the label of any of M's instructions. *M's configuration* where M is in state m, and k_1 and k_2 are the current values of counters r_1 and r_2, respectively, is denoted by $(m; k_1, k_2)$. A *computation* performed by M is a sequence of M's configurations such that each step is made by one of the above instructions: $(a_1; n, 0) \xrightarrow{a_1} \cdots \longrightarrow (a_i; k_1, k_2) \xrightarrow{a_i} (b_j; k_1', k_2') \xrightarrow{b_j} \cdots$ A *terminating* computation is one that ends in a configuration $(a_0; *, *)$, that is, the final state a_0 with any values in the counters.

Furthermore, for encoding purposes, each instruction of the form a_i : **if** $(r_1 = 0)$ **goto** b_j **else goto** b_k; is replaced by the following set of instructions:

$$\begin{array}{l} a_i : \textbf{ if } (r_1 = 0) \textbf{ goto } b_{\hat{\jmath}} \textbf{ else goto } b_k; \\ b_{\hat{\jmath}} : \textbf{ goto } a_{\hat{\imath}}; \\ a_{\hat{\imath}} : \textbf{ goto } b_j; \end{array} \tag{1}$$

and each instruction of the form b_j : **if** $(r_2 = 0)$ **goto** a_k **else goto** a_l; is replaced by the following set of instructions:

$$\begin{array}{l} b_j : \textbf{ if } (r_2 = 0) \textbf{ goto } a_{\tilde{k}} \textbf{ else goto } a_l; \\ a_{\tilde{k}} : \textbf{ goto } b_{\tilde{\jmath}}; \\ b_{\tilde{\jmath}} : \textbf{ goto } a_k; \end{array} \tag{2}$$

where $b_{\hat{\jmath}}$, $a_{\hat{\imath}}$, $a_{\tilde{k}}$, $b_{\tilde{\jmath}}$ are fresh, unique labels. It is important to note that instructions labeled by $b_{\hat{\jmath}}$ and $a_{\hat{\imath}}$ can be applied only in a row, and only in the case when the corresponding value of counter r_1 is zero. Similarly instructions labeled by $a_{\tilde{k}}$ and $b_{\tilde{\jmath}}$ can be applied only in a row, and only in the case when the corresponding value of counter r_2 is zero.

5.2 Minsky Machine Translation

We can now describe how to embed a given Minsky machine M into a local state transition system which we denote

T_M. The system T_M will have two participants, Alice and Bob. They will each limit their own local configuration to contain only one fact. Alice will use public tokens to track the value of register 1, and Bob will similarly track the value of register 2.

States of agent A are encoded by private propositions: r^A, s_0^A, s_1^A,.., s_i^A,.... States of agent B are encoded by private propositions: r^B, s_0^B, s_1^B,.., s_j^B,.... A public proposition R_1 means that "One public resource unit is allocated to Alice." A public proposition R_2 means that "One public resource unit is allocated to Bob." A public predicate $C_1(x)$ means "x is the label of an instruction to be run by Alice". A public predicate $C_2(x)$ means "x is the label of an instruction to be run by Bob". M's configuration of the form $(a_i; k_1, k_2)$ is encoded by $r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_1(a_i)$. M's configuration of the form $(b_j; k_1, k_2)$ is encoded by $r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_2(b_j)$.

Alice's actions roughly correspond to M's instructions on register one. For every instruction of the form a_i : **goto** b_j; Alice has the corresponding jump action:

$$s_i^A \quad \rightarrow_A \quad r^A, C_2(b_j). \tag{3}$$

This action does not change the public resources, and it writes the public fact $C_2(b_j)$ to tell Bob which instruction to perform next. Similarly, for every instruction of the form $a_i : r_1 := r_1+1$; **goto** b_j; Alice has the corresponding addition action:

$$s_i^A \quad \rightarrow_A \quad r^A, R_1, C_2(b_j). \tag{4}$$

For every instruction of the form $a_i : r_1 := r_1-1$; **goto** b_j; Alice has the corresponding subtraction action:

$$s_i^A, R_1 \quad \rightarrow_A \quad r^A, C_2(b_j). \tag{5}$$

For every instruction of the form a_i : **if** $(r_1 = 0)$ **goto** $b_{\hat{\jmath}}$ **else goto** b_k; Alice has two actions:

$$s_i^A \quad \rightarrow_A \quad r^A, C_2(b_{\hat{\jmath}}) \tag{6}$$

$$s_i^A, R_1 \quad \rightarrow_A \quad r^A, R_1, C_2(b_k). \tag{7}$$

Notice that we introduce nondeterminism here. Without any way of controlling this nondeterminism, an agent may use action (6) when there is an instance of R_1. The machine M cannot take this direction because M first checks if $r_1 = 0$ and if not, it follows the other branch of the instruction. The use of critical configurations play an essential role in controlling this nondeterminism as we discuss below. Also note that although actions of types (3) and (6) have a similar form, they are differentiated syntactically by their labels. In particular, actions of type (6) publish one of the distinguished labels $b_{\hat{\jmath}}$ that were introduced in construction (1).

Finally, Alice has an action that does not directly correspond to any of M's instructions. This action explicitly links the labels a_i with the corresponding propositions s_i^A. For each corresponding pair, Alice has the receive action

$$r^A, C_1(a_i) \quad \rightarrow_A \quad s_i^A \tag{8}$$

in which Alice reads the label of an instruction from the network and enters a state ready to perform that instruction.

Bob has actions on his side that correspond to instructions on M's register 2. For completeness, we list those actions in the corresponding order below.

$$s_j^B \quad \rightarrow_B \quad r^B, C_1(a_k) \tag{9}$$

$$s_j^B \quad \rightarrow_B \quad r^B, R_2, C_1(a_k) \tag{10}$$

$$s_j^B, R_2 \quad \rightarrow_B \quad r^B, C_1(a_k) \tag{11}$$

$$s_j^B \quad \rightarrow_B \quad r^B, C_1(a_{\tilde{k}}) \tag{12}$$

$$s_j^B, R_2 \quad \rightarrow_B \quad r^B, R_2, C_1(a_l) \tag{13}$$

$$r^B, C_2(b_j) \quad \rightarrow_B \quad s_j^B \tag{14}$$

Thus associated to any Minsky machine M is the local state transition system T_M with actions of types (3)–(14). The initial configuration of M is $(a_1; n, 0)$ and the corresponding initial configuration of T_M is $r^A, R_1^n, r^B, C_1(a_1)$.

In order to fully specify the problem of finding a compliant or a weakly compliant plan we must describe what the goal configurations and confidentiality policies are. We specify a single goal of $C_1(a_0)$. This simply corresponds to an accepting state of M. Thus any configuration containing $C_1(a_0)$ is a goal configuration. We need to take some care in choosing what confidentiality policies to use because they will be used to control the potentially undesirable behavior caused by the nondeterminism introduced in the translation of the 0-test instructions. If we do not specify any policies (or specify the empty policies) then the translation will not be faithful. That is, a plan might follow action (6) when there is an instance of R_1.

We choose critical configurations which designate this behavior as undesirable. Namely, Alice's critical configurations are those of the form $C_1(a_{\hat{\imath}}), R_1$ and Bob's critical configurations are those of the form $C_2(b_{\hat{\jmath}}), R_2$ where $a_{\hat{\imath}}$ and $b_{\hat{\jmath}}$ are the fresh unique labels from constructions (1) and (2) respectively. Intuitively, since the labels $a_{\hat{\imath}}$ and $b_{\hat{\jmath}}$ should only occur when $r_1 = 0$ and $r_2 = 0$ respectively, these critical configurations serve as a signal that the wrong branch was taken in a plan. Thus if compliant plans and weakly compliant plans are supposed to accurately represent machine computations, they should never reach these configurations.

Notice that these critical configurations do not take full advantage of the expressive power of confidentiality policies. In particular they consist only of public facts. This means that both Alice and Bob can recognize when a configuration is critical for either agent. We show undecidability even in this more restrictive case.

It remains to show that this translation allows T_M to faithfully simulate terminating computations of M. The next subsection provides the necessary soundness and completeness results.

5.3 Soundness and Completeness

In this section we show that, under our translation of Minsky machines into LSTSs (with confidentiality policies), the machine M has a terminating computation on $(a_1; n, 0)$ leading to $(a_0; *, *)$ if and only if T_M has a weakly compliant plan leading from $r^A, R_1^n, r^B, C_1(a_1)$ to $C_1(a_0)$ if and only if T_M has a compliant plan leading from $r^A, R_1^n, r^B, C_1(a_1)$ to $C_1(a_0)$.

We first show the soundness of our translation with respect to plan compliance. Since plan compliance implies weak plan compliance the corresponding soundness result with respect to weak plan compliance is an immediate corollary. We then show completeness of the translation with respect to weak plan compliance. Again, since plan compliance implies weak plan compliance, we get the completeness with respect to plan compliance as an immediate corollary.

Proposition 3 (Soundness) *Given a Minsky machine M and its translation T_M, if M's computation on $(a_1; n, 0)$ terminates in $(a_0; *, *)$, then T_M has a compliant plan leading from $r^A, R_1^n, r^B, C_1(a_1)$ to the partial goal $C_1(a_0)$.*

Proof. We show that whenever M and T_M are in corresponding configurations, if M's instructions lead to a configuration of the form $(s_i; k_1, k_2)$, then T_M can reach the corresponding configuration which represents $(s_i; k_1, k_2)$. We then show that the resulting plan is compliant. The argument proceeds by induction on the length of the M-computation with the basis step being an M-computation of only 1 step. (The 0-step case is trivial.)

Basis Step: $(a_i; k_1, k_2) \rightarrow_M^1 (b_j; k_1', k_2')$

We examine each possible instruction that M might perform.

Case: a_i: **goto** b_j*;*

M goes from $(a_i; k_1, k_2)$ to $(b_j; k_1, k_2)$. Then T_M satisfies

$$r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_1(a_i) \quad \triangleright^* \quad r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_2(b_j)$$

by applying a receive action of type (8) followed by a jump action of type (3). Notice that the configuration on the right represents the configuration $(b_j; k_1, k_2)$ as desired.

Case: $a_i : r_1 := r_1 + 1$*;* **goto** b_j*;*

M goes from $(a_i; k_1, k_2)$ to $(b_j; k_1 + 1, k_2)$. Then T_M satisfies

$$r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_1(a_i) \quad \triangleright^* \quad r^A, R_1^{k_1+1}, r^B, R_2^{k_2}, C_2(b_j)$$

by applying a receive action of type (8) followed by an addition action of type (4). Notice that the configuration on the right represents the configuration $(b_j; k_1 + 1, k_2)$ as desired.

Case: $a_i : r_1 := r_1 - 1$; **goto** b_j*;*

M goes from $(a_i; k_1, k_2)$ to $(b_j; k_1 - 1, k_2)$. Since M can perform this instruction we know that $k_1 > 0$. Thus T_M satisfies

$$r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_1(a_i) \quad \triangleright^* \quad r^A, R_1^{k_1-1}, r^B, R_2^{k_2}, C_2(b_j)$$

by applying a receive action of type (8) followed by a subtraction action of type (5) (which is applicable because $k_1 > 0$). Notice that the configuration on the right represents the configuration $(b_j; k_1 - 1, k_2)$ as desired.

Case: a_i: **if** $(r_1 = 0)$ **goto** $b_{\widehat{j}}$ **else goto** b_k;

M goes from $(a_i; k_1, k_2)$ to either $(b_{\widehat{j}}; k_1, k_2)$ if $k_1 = 0$ or $(b_k; k_1, k_2)$ if $k_1 > 0$. In the first case T_M satisfies

$$r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_1(a_i) \quad \triangleright^* \quad r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_2(b_{\widehat{j}})$$

by applying a receive action of type (8) followed by an action of type (6). In the latter case, T_M satisfies

$$r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_1(a_i) \quad \triangleright^* \quad r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_2(b_k)$$

by applying a receive action of type (8) followed by an action of type (7) (which is enabled because $k_1 > 0$ in this case). Notice that the configurations on the right represent $(b_{\widehat{j}}; k_1, k_2)$ and $(b_k; k_1, k_2)$ respectively as desired.

We conclude that any single instruction on M's register 1 can be simulated by T_M's actions. As stated above, the cases in which M's instruction modifies register 2 are completely analogous, and so we omit the details here.

Induction Step: This follows immediately from the inductive hypothesis by splitting a length n computation into a length $n - 1$ computation followed by a length 1 computation. Therefore computations of any length can be simulated by T_M. In particular, if $(a_1; n, 0) \rightarrow_M^* (a_0; *, *)$ then we can conclude that T_M satisfies

$$r^A, R_1^n, r^B, C_1(a_1) \quad \rightsquigarrow^* \quad C_1(a_0).$$

We must now show why this plan is compliant. Recall that Alice's critical configurations are of the form $C_1(a_{\widehat{i}}), R_1$ where $a_{\widehat{i}}$ is the label which appears only in an instruction sequence resulting from the transformation (1). We first show why none of the configurations in the plan are critical. Every configuration of the form $r^A, R_1^{k_1}, r^B, R_2^{k2}, C_1(a_{\widehat{i}})$ in the plan corresponds to a configuration $(a_{\widehat{i}}; k_1, k_2)$ along M's computation. We remarked at the end of Section 5.1, that

this only happens when $k_1 = 0$. Thus none of the configurations of the plan are critical for Alice. A similar argument shows none of the configurations are critical for Bob.

Now suppose that Bob tries to deviate from the plan to create a configuration which is critical for Alice. Without loss of generality we may assume he starts from a configuration of the form $r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_2(b_j)$.

He must create a fact $C_1(a_{\hat{i}})$ (with a distinguished label) in order to succeed. By inspection we can see that Bob can only do this if $b_j = b_{\hat{j}}$. But that means that the machine configuration is $(b_{\hat{j}}; k_1, k_2)$ which we already remarked happens only when $k_1 = 0$. Thus, if Bob succeeds in creating the fact $C_1(a_{\hat{i}})$ it means that there are no instances of R_1 in the configuration, and the result is not critical for Alice. A similar argument shows that Alice cannot create a critical configuration for Bob by herself, if she starts from a configuration in the plan. ■

Thus the translation is sound with respect to both plan compliance and weak plan compliance. In order to show that the translation is complete, we show that an arbitrary weakly compliant plan leading from $r^A, R_1^n, r^B, C_1(a_1)$ to $C_1(a_0)$ corresponds to the terminating computation of M on $(a_1; n, 0)$. This direction requires some care because we do not know *a priori* that a weakly compliant plan has a form which clearly corresponds to a computation of M. Thus we first show that weakly compliant plans do have a nice form passing only through configurations which we call regular.

Definition 9 *If T_M is the translation of a Minsky machine M, then a configuration is called* regular *if it has one of the four following forms:*

$$r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_1(a_i) \quad \textit{(I)}$$

$$s_i^A, R_1^{k_1}, r^B, R_2^{k_2} \quad \textit{(II)}$$

$$r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_2(b_j) \quad \textit{(III)}$$

$$r^A, R_1^{k_1}, s_j^B, R_2^{k_2} \quad \textit{(IV)}$$

Configurations of type (I) and (III) correspond directly to Minsky machine configurations as we saw before. Configurations of type (II) and (IV) are the "intermediate" configurations that arise from the use of the receive actions (8) and (14). In the previous proof, every configuration of the plan was regular. The next lemma states that this must be the case for every plan which starts in a regular configuration.

Lemma 1 *Let T_M be the translation of a Minsky machine M. If T_M begins in a regular configuration then every configuration in a plan is also regular. Furthermore, the configurations cycle through types (I), (II), (III) and (IV) in that order.*

Proof. The full proof requires a case analysis on each type of regular configuration. For example, if a system in a configuration of type (I), then by inspection of the actions, we see that any action which is enabled will transform the configuration into a configuration of type (II). The other cases are proved by a similar inspection of the actions which we omit here for reasons of space. ■

Using this result we can now show how to simulate an arbitrary weakly compliant plan by a computation of M.

Proposition 4 (Completeness) *Given a Minsky machine M and its translation T_M, if T_M has a weakly compliant plan leading from $r^A, R_1^n, r^B, C_1(a_1)$ to $C_1(a_0)$ then M's computation leads from $(a_1; n, 0)$ to $(a_0; *, *)$.*

Proof. We prove more generally that if the weakly compliant plan leads from $r^A, R_1^n, r^B, C_1(a_1)$ to $r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_1(a_i)$ then M's computation leads from $(a_1; n, 0)$ to $(a_i; k_1, k_2)$. Since $r^A, R_1^n, r^B, C_1(a_1)$ is a regular configuration, Lemma 1 implies that the plan proceeds to enter a cycle of regular configurations. Since a regular configuration which contains $C_1(a_i)$ is of type (I) we know that the plan must start and end in a configuration of type (I) and therefore must consist of some number of complete cycles (and no partial cycles). We prove the result by induction on the number of cycles in the plan. Recall that a full cycle consists of four actions, two actions from Alice followed by two actions from Bob.

Basis Step: 0 Cycles

If the plan has 0 cycles then it starts in the ending state and hence $a_1 = a_i$, $n = k_1$ and $k_2 = 0$ so M also starts in the configuration $(a_i; k_1, k_2)$.

Induction Step:

The inductive hypothesis is that a weakly compliant plan with $n-1$ cycles can be simulated by an M-computation. We must then show that an n-cycle plan can also be simulated by an M-computation.

After the first $n-1$ cycles of the plan, T_M is in a regular configuration of type (I) that looks like the following:

$$r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_1(a_i)$$

By the inductive hypothesis we know that M leads to the corresponding configuration $(a_i; k_1, k_2)$. We must only show how M simulates the last cycle of the plan. By Lemma 1 we know that the cycle begins with a receive action of type (8) followed by an action of one of the types (3)–(7).

Case I:

Alice applies a receive action followed by an action of type (3). This transforms T_M into a configuration of the form

$$r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_2(b_j).$$

Since the second action is of type (3), the label a_i is the label for a jump instruction, and M can apply the instruction and end up in the configuration $(b_j; k_1, k_2)$ which is represented by T_M's resulting configuration.

Case II:

Alice applies a receive action followed by an action of type (4). This transforms T_M into a configuration of the form

$$r^A, R_1^{k_1+1}, r^B, R_2^{k_2}, C_2(b_j).$$

Since the second action is of type (4), the label a_i is the label for an addition instruction, and M can apply the instruction and end up in the configuration $(b_j; k_1 + 1, k_2)$ which is represented by T_M's resulting configuration.

Case III:

Alice applies a receive action followed by an action of type (5). This transforms T_M into a configuration of the form

$$r^A, R_1^{k_1-1}, r^B, R_2^{k_2}, C_2(b_j).$$

Since the second action is of type (5), the label a_i is the label for a subtraction instruction. Since T_M's action requires the existence of an occurrence of R_1 we are guaranteed that $k_1 > 0$. Thus M can successfully apply the instruction and end up in the configuration $(b_j; k_1 - 1, k_2)$ which is represented by T_M's resulting configuration.

Case IV:

Alice applies a receive action followed by an action of type (6). This transforms T_M into a configuration of the form

$$r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_2(b_{\widehat{j}}).$$

Since the second action is of type (6), the label a_i is the label for a 0-test instruction. Furthermore, we know there are no occurrences of R_1 because the plan is assumed to be weakly compliant. An occurrence of R_1 would cause the cycle to finish with Bob publishing the fact $C_1(a_{\widehat{i}})$ creating a critical configuration contrary to the weak compliance of the plan. Thus $k_1 = 0$ and M's previous configuration is actually $(a_i; 0, k_2)$. Hence M's test for 0 in register 1 is successful. M then ends up in the configuration $(b_{\widehat{j}}; 0, k_2)$ which is represented by T_M's resulting configuration.

Case V:

Alice applies a receive action followed by an action of type (7). This transforms T_M into a configuration of the form

$$r^A, R_1^{k_1}, r^B, R_2^{k_2}, C_2(b_k).$$

Since the second action is of type (7), the label a_i is the label for a 0-test instruction. Furthermore, we know $k_1 > 0$ because the action is only enabled when there is an occurrence of R_1. Thus when M performs the 0-test, it finds $k_1 > 0$ and transitions into the configuration $(b_k; k_1, k_2)$ which is represented by T_M's resulting configuration.

The cycle is completed by Bob applying a receive action of type (14) followed by an action of one of the types (9)–(13). The analysis of each of these cases is completely analogous to the previous cases completing the induction. We therefore conclude that M can simulate any weakly compliant plan which completes some number of full cycles. In particular, if the weakly compliant plan leads from $r^A, R_1^n, r^B, C_1(a_1)$ to $C_1(a_0)$, then M's computation leads from $(a_1; n, 0)$ to $(a_0; *, *)$. ■

Since every compliant plan is also weakly compliant, it follows that the translation is complete with respect to plan compliance as well. The soundness and completeness results combine to imply that Problem 2 (weak plan compliance) and Problem 3 (plan compliance) are both undecidable.

Theorem 2 (Weak Plan Compliance) *Let T be a local state transition system with a finite set of propositional actions, $\mathcal{Z}$ a finite set of goal configurations, $\mathcal{C}$ a finite set of critical configurations, and W an initial configuration. The problem of determining if T has a weakly compliant plan leading from W to one of the goal configurations is undecidable.*

Proof. The translation from Minsky machines creates a local state transition system T_M with only finitely many propositional actions, a single goal configuration, and a finite set of critical configurations. The soundness and completeness of the translation imply the undecidability of the problem. ■

Theorem 2 is in contrast with the PSPACE result of [21] for weak plan compliance when the actions are restricted to be well-balanced (see Figure 1). Allowing un-balanced actions alters the decidability of the problem. Also note that in the well-balanced case, system compliance and weak plan compliance have the same complexity, but in the general case system compliance is decidable and weak plan compliance becomes undecidable. Therefore, the undecidability arises from a combination of the presence of un-balanced actions and the importance of current configurations as embodied by the definition of weak plan compliance.

Theorem 3 (Plan Compliance) *Let T be a local state transition system with a finite set of propositional actions, $\mathcal{Z}$ a finite set of goal configurations, $\mathcal{C}$ a finite set of critical configurations, and W an initial configuration. The problem of determining if T has a compliant plan leading from W to one of the goal configurations is undecidable.*

Proof. This proof is identical to the proof of Theorem 3. ■

We also want to point out the differences between Theorems 2 and 3 and the undecidability results of [13] and [25]. The results of [13] are about secrecy in the context of security protocols with an external intruder who has considerably more power than the honest protocol participants. In that setting, undecidability is only attained with the ability to create an unbounded number of fresh values, or nonces, combined with the unbounded memory of the intruder. In the current work, we take advantage of the unbounded memory of the agents by allowing un-balanced actions, but we work in a propositional setting in which no new values are ever created.

Similarly, the proof of undecidability for pure monadic linear logic [25] relies on the ability to use the existential quantifier to create an unbounded number of new values. Also, linear logic derivability roughly correspond to exact reachability in our setting. We lose some control by using our weaker notion of reachability, and by not allowing new values to be created, but we regain that control through the interpretation of confidentiality policies.

6 Related Work

We are certainly not the first to consider security in a collaborative setting. For example, other authors have analyzed privacy aspects of algorithms designed to collaboratively schedule meetings as well as other related distributed constraint optimization problems [19, 18]. Instead of focusing on policy compliance, these papers examine several metrics to measure the amount of privacy lost in various algorithms.

In [1, 10] the authors present a model that accounts for three types of participants: Byzantine, altruistic and rational. This is a framework meant to design and analyze protocols and processes in which some agents act in their own self interest, some can act in arbitrary ways, and some follow a deterministic set of instructions. The concept is general, but the incentives depend on the specific instance. Analyses are carried out on a case by case basis. At a more abstract level, [39] characterizes boolean functions that can be non-cooperatively computed. These are functions for which the participants will honestly state their input values to a central, trusted algorithm, assuming the agents care more about learning the outcome of the function than the input values of other agents. All of these papers incorporate explicit utility functions. While these differ from our confidentiality policies in detail, they represent similar confidentiality constraints that must be satisfied by "successful" executions.

A general, formal model for privacy is presented in [4, 5] called Contextual Integrity. The authors focus on more than just confidentiality. They consider more general norms for the appropriate transmission of information. They consider both positive and negative norms which respectively generalize "allow" and "deny" rules in traditional access control. As the name suggests, the authors also emphasize the importance of context in determining which norms are applicable in a given situation. This determination may be imposed externally by law, or internally in the form of company privacy policies. Indeed the importance of context was brought to our attention by these papers.

While our notion of confidentiality depends on reaching a particular configuration or not, other notions have been considered which focus on an adversary's ability to distinguish between two traces. For example [2, 3] define secrecy in this way. This is very much in the spirit of non-interference [32, 35, 17, 31] in which secret, or High, variables should not leak to public, or Low, variables. These distinctions tend to be static and absolute, which makes them inflexible and hard to adapt to different contexts. Furthermore, the focus is on the past value of certain variables, while our confidentiality policies have the ability to express knowledge of the *current* configuration. Declassification must be handled very carefully in this setting [32, 36, 33].

We also point out that there may be strong connections between our complexity and undecidability results, and similar results from AI planning. The complexity of the planning problem has been shown to be sensitive to changes in the details of the problem statement [14, 9] regarding the use of function symbols, deletion lists and negative pre-conditions. We have not explored the possible connections and equivalences between the problems of this work and problems in classical planning. In a similar vein, the authors of [37] modify linear logic to incoporate constraints that correspond to the more continuous aspects of traditional planning problems that one encounters in the real world.

Our formalism also shares some similarities with the Dolev-Yao model of security protocols [12, 13] in which a powerful intruder tries to disrupt communication between honest protocol participants. Our framework differs from the Dolev-Yao model by considering evenly matched agents who can loop and who may have unbounded memory. Also, agents in the Dolev-Yao model can create fresh values or nonces, whereas we have not yet provided for that possibility in our framework. Our model is closer in spirit to the Dolev-Yao model of contract signing protocols [7, 8] if *all* participants are "optimistic".

7 Conclusion and Future Work

In this paper we have explored three notions of policy compliance on top of the underlying model of local state transition systems. Those three interpretations are system compliance, weak plan compliance, and plan compliance. Since the level of trust among collaborating agents may differ depending on the agents and the context, each interpretation of the agents' confidentiality policies is appropriate for a different level of trust among the agents. In contrast to our previous work, our current focus was on determining

the decidability and complexity of policy compliance and goal reachability in the general case with *un-balanced actions.* We showed that the collaborative planning problem with system compliance is EXPSPACE-complete. Also, the collaborative planning problem with weak plan compliance and the collaborative planning problem with plan compliance are both undecidable. This undecidability is due to the importance of the current configuration of the system in both versions of plan compliance. This is in contrast with other undecidability results in the security literature [13] which require the creation of fresh values.

Looking towards possible directions of future research, we note that the computational problems considered in this paper require full information about each agent's actions and policies. As a participant in the collaboration, however, one would like to be able to make some local assessment of the confidentiality guarantees provided by the system. One advantage to this approach is that each agent could choose the interpretation of their confidentiality policy that best suits their needs. It may be that methods from mechanism design can help to decentralize the analysis. Such an approach is likely to entail the introduction of explicit utility functions and incentives to the model. This may be the right setting in which to explore connections between the definition of plan compliance and types of Nash equilibria as mentioned in Section 3.

Some other directions aim at increasing the expressibility of the model. Many algorithms and protocols require the use of random values or fresh values. In the past, the use of the existential quantifier has proven successful as a tool for modeling them [13]. We would like to explore the complexity of the three problems discussed in this paper, in both the unbalanced and well-balanced cases, when existentials are used.

It would be interesting to consider lifting some of the restrictions about the locality of the actions. Namely, we might consider restrictions that correspond to "read" and "write" privileges with the structure of a lattice [11]. The extra structure could be potentially useful for more natural models of communication and for expressing even richer confidentiality policies.

The confidentiality policies we use in this paper do not place any explicit restrictions on the actions. Many real-world policies have rules about when an agent may or must perform an action and when an action is prohibited. We may be able to express such policies by imposing parametric conditions on the actions, including boolean conditions. This would allow the model to align more closely with real-world scenarios. Similarly, we might be able to introduce parameters that help determine the context. Currently each context has a corresponding transition system. Parameters could determine which actions are available and which configurations are critical, allowing the model to consider the impact on an agent of changing contexts in various ways.

Finally, time can be an important factor in some places like financial institutions. Using an asynchronous model such as the one in this paper is not well-suited to these time-sensitive situations. We want to explore a similar synchronous model which has some notion of explicit time. We believe this could provide a rich exploration of confidentiality concerns which are not expressible in the current model.

Acknowledgements

Thanks to Anupam Datta, George Dinolt, Rachel Greenstadt, Joshua Guttman, Pat Lincoln, John Mitchell, Jose Meseguer, Helen Nissenbaum, Tim Roughgarden, Natarajan Shankar and Paul Syverson for their very helpful discussions and comments.

Scedrov and Rowe were partially supported by ONR Grant N00014-07-1-1039, by OSD/AFOSR MURI "Collaborative policies and assured information sharing", and by NSF Grants CNS-0429689, CNS-0524059, and CNS-0830949. Kanovich was partially supported by EPSRC Grant EP/D053625/1 "Modularity and Resource Separation" and by ONR Grant N00014-07-1-1039. This material is based upon work supported by the MURI program under AFOSR Grant No: FA9550-08-1-0352

References

[1] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. Bar fault tolerance for cooperative services. *SIGOPS Oper. Syst. Rev.*, 39(5):45–58, 2005.

[2] R. Alur, P. Černý, and S. Chaudhuri. Model checking on trees with path equivalences. In *TACAS 2007*, pages 664–678. Springer, 2007.

[3] R. Alur, P. Černý, and S. Zdancewic. Preserving secrecy under refinement. In *ICALP '06: Proceedings (Part II) of the 33rd International Colloquium on Automata, Languages and Programming*, pages 107–118. Springer, 2006.

[4] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. Privacy and contextual integrity: Framework and applications. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 184–198. IEEE Computer Society, 2006.

[5] A. Barth, A. Datta, J. C. Mitchell, and S. Sundaram. Privacy and utility in business processes. In *Proceedings of 20th IEEE Computer Security Foundations Symposium*, July 2007.

[6] W. Bibel. A deductive solution for plan generation. *New Generation Computing*, 4(2):115–132, 1986.

[7] R. Chadha, M. I. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In *ACM Conference on Computer and Communications Security*, pages 176–185, 2001.

[8] R. Chadha, J. C. Mitchell, A. Scedrov, and V. Shmatikov. Contract signing, optimism, and advantage. *J. Log. Algebr. Program.*, 64(2):189–218, 2005.

[9] D. Chapman. Planning for conjunctive goals. Technical report, Cambridge, MA, USA, 1985.

[10] A. Clement, H. C. Li, J. Napper, J.-P. Martin, L. Alvisi, and M. Dahlin. Bar primer. In *DSN*, pages 287–296, 2008.

[11] D. E. Denning. A lattice model of secure information flow. *Commun. ACM*, 19(5):236–243, 1976.

[12] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[13] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[14] K. Erol, D. S. Nau, and V. S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. *Artif. Intell.*, 76(1-2):75–88, 1995.

[15] V. Gehlot and C. Gunter. Normal process representatives. In *Proc. of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 200–207, Philadelphia, PA, 1990.

[16] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[17] J. A. Goguen and J. Meseguer. Security policies and security models. In *IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

[18] R. Greenstadt, J. P. Pearce, and M. Tambe. Analysis of privacy loss in distributed constraint optimization. In *AAAI*, Boston, MA, July 2006.

[19] R. Greenstadt and M. D. Smith. Collaborative scheduling: Threats and promises. In *Proceedings of the Fifth Annual Workshop on Economics and Information Security*, Cambridge, England, June 2006.

[20] J. Hopcroft and J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Thoeretical Computer Science*, 8(2):135–159, 1979.

[21] M. Kanovich, P. Rowe, and A. Scedrov. Collaborative planning with confidentiality. *Submitted*. `http://www.math.upenn.edu/~rowep/CPWC_full.pdf`.

[22] M. Kanovich, P. Rowe, and A. Scedrov. Collaborative planning with privacy. In *20th IEEE Computer Security Foundations Symposium (CSF 20)*, Venice, Italy, July 2007.

[23] M. Kanovich and J. Vauzeilles. The classical AI planning problems in the mirror of Horn linear logic: semantics, expressibility, complexity. *Mathematical Structures in Computer Science*, 11(6):689–716, 2001.

[24] M. I. Kanovich. Petri nets, Horn programs, linear logic and vector games. *Annals of Pure and Applied Logic*, 75(1-2):107–135, 1995.

[25] M. I. Kanovich. The expressive power of horn monadic linear logic. In *CSL*, pages 39–53, 2001.

[26] R. M. Karp and R. E. Miller. Parallel program schemata. *Journ. Computer and System Sciences*, 3(2):147–195, May 1969.

[27] A. P. Kopylov. Decidability of linear affine logic. In *LICS '95: Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science*, page 496, Washington, DC, USA, 1995. IEEE Computer Society.

[28] P. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar. Decision problems for propositional linear logic. *Annals Pure Applied Logic*, 56:239–311, 1992.

[29] R. J. Lipton. The reachability problem requires exponential space. Technical report, Department of Computer Science, Research Report 62, Yale University, 1976.

[30] D. McDermott and J. Hendler. Planning: What it is, what it could be, An introduction to the special issue on planning and scheduling. *Artificial Intelligence*, 76:1–16, 1995.

[31] J. McLean. Security models. In J. Marciniak, editor, *Encyclopedia of Software Engineering*. John Wiley & Sons, 1994.

[32] A. C. Myers, A. Sabelfeld, and S. Zdancewic. Enforcing robust declassification and qualified robustness. *Journal of Computer Security*, 14(2):157–196, 2006. Extended abstract in CSFW pages 172-186, 2004.

[33] S. Pinsky. Absorbing covers and intransitive noninterference. In *SP '95: Proceedings of the 1995 IEEE Symposium on Security and Privacy*, page 102, Washington, DC, USA, 1995. IEEE Computer Society.

[34] C. Rackoff. The covering and boundedness problem for vector addition systems. *Theoretical Computer Science*, 6:223–231, 1978.

[35] J. C. Reynolds. Syntactic control of interference. In *Symposium on Principles of Programming Languages (POPL)*, pages 39–46, 1978.

[36] A. W. Roscoe. What is intransitive noninterference. In *In Proc. of the 12th IEEE Computer Security Foundations Workshop*, pages 228–238, 1999.

[37] U. Saranli and F. Pfenning. Using constrained intuitionistic linear logic for hybrid robotic planning problems. In *ICRA*, pages 3705–3710, 2007.

[38] J. E. Savage. *Models of Computation: Exploring the Power of Computing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[39] Y. Shoham and M. Tennenholtz. Non-cooperative computation: boolean functions with correctness and exclusivity. *Theor. Comput. Sci.*, 343(1-2):97–113, 2005.

[40] H.-C. Yen. Introduction to Petri net theory. *Recent Advances in Formal Languages and Applications*, 25:343–373, 2006.

A Reductions Between Petri nets and LSTSs

In this appendix we give the details of the correspondence between the Petri net coverability problem and the LSTS reachability problem.

A.1 Reduction from Petri nets to LSTSs

Let $N = (P, T, \phi)$ be a Petri net. We will associate a LSTS T_N to this Petri net. For each place p_i we will include a propositional constant (0-ary predicate) F_i in the signature of T_N. For any marking μ of N we associate the configuration

$$\hat{\mu} = F_1^{\mu(p_1)}, \ldots, F_k^{\mu(p_k)}.$$

For each transition t_i we include the (propositional) action

$$\hat{t}_i = F_1^{\phi(p_1,t_i)}, \ldots, F_k^{\phi(p_k,t_i)} \to F_1^{\phi(t_i,p_1)}, \ldots, F_k^{\phi(t_i,p_k)}.$$

This action is enabled in a configuration $\hat{\mu}$ iff $\hat{\mu}(p_j) \geq \phi(p_j, t_i)$ for all $1 \leq j \leq k$ iff t_i is enabled in μ. Thus the result of applying this action to the configuration $\hat{\mu}$ is

$$\hat{\mu}' = F_1^{\mu(p_1)-\phi(p_1,t_i)+\phi(t_i,p_1)}, \ldots, F_k^{\mu(p_k)-\phi(p_k,t_i)+\phi(t_i,p_k)}$$

where $\mu \xrightarrow{t_i} \mu'$.

Note that T_N has finitely many propositional actions. Notice also that the size of the LSTS reachability problem (as measured by the total length of the binary descriptions of the actions, and the initial and final configurations) is of the same order as the size of the Petri net coverability problem (measured by the total length of the binary descriptions of the flow function and the initial and final markings). Under appropriate binary encodings this translation can be performed in polynomial time.

Proposition 5 *Under the above translation, $\mu_1 \in C(N, \mu_0)$ if and only if $\hat{\mu}_0 \rightsquigarrow^*_{T_N} \hat{\mu}_1$.*

Proof. We start with the forward direction. The proof is by induction on the length of the firing sequence σ which satisfies $\mu_0 \xrightarrow{\sigma} \mu$ such that $\mu \geq \mu_1$. The basis case of length 0 is trivial.
Induction Step:

Suppose that if μ is coverable from μ_0 via a firing sequence of length n then $\hat{\mu}_0 \rightsquigarrow^*_{T_N} \hat{\mu}$. Suppose also that μ_1 is coverable from μ_0 via a firing sequence of length $n+1$. Then there are markings μ, μ' such that $\mu_0 \xrightarrow{\sigma} \mu \xrightarrow{t_i} \mu'$ where σ is a firing sequence of length n and $\mu' \geq \mu_1$. By the inductive hypothesis, $\hat{\mu}_0 \rightsquigarrow^*_{T_N} \hat{\mu}$. It is easy to check that since $\mu \xrightarrow{t_i} \mu'$, we also get that $\hat{\mu} \rhd^1_{t_i} \hat{\mu}'$. Also since $\mu' \geq \mu_1$ we know that $\hat{\mu}'$ contains $\hat{\mu}_1$. Thus, $\hat{\mu} \rightsquigarrow^*_{T_N} \hat{\mu}_1$. It is easily seen that $\rightsquigarrow^*_{T_N}$ is a transitive relation, thus $\hat{\mu}_0 \rightsquigarrow^*_{T_N} \hat{\mu}_1$ as desired. This completes the proof of the forward direction.

We now look at the other direction. The proof is by induction on the length of the sequence of actions which witnesses $\hat{\mu}_0 \rightsquigarrow^*_{T_N} \hat{\mu}_1$. The basis case of a length 0 sequence is trivial.
Induction Step:

Suppose that if $\hat{\mu}_0 \rightsquigarrow^n_{T_N} \hat{\mu}$ then $\mu \in C(N, \mu_0)$. Suppose also that $\hat{\mu}_0 \rightsquigarrow^{n+1}_{T_N} \hat{\mu}_1$. Then there is a configuration $\hat{\mu}$ such that $\hat{\mu}_0 \rightsquigarrow^n_{T_N} \hat{\mu} \rightsquigarrow^1_{\hat{t}_i} \hat{\mu}_1$. By the inductive hypothesis we know that $\mu \in C(N, \mu_0)$. Also, by definition, there is a configuration $\hat{\mu}'$ such that $\hat{\mu} \rhd^1_{\hat{t}_i} \hat{\mu}'$ and $\hat{\mu}'$ contains $\hat{\mu}_1$. It is easy to check that $\mu \xrightarrow{t_i} \mu'$ and $\mu' \geq \mu_1$. Thus $\mu_1 \in C(N, \mu)$. It is not too difficult to see that coverability is transitive and so $\mu_1 \in C(N, \mu_0)$ as desired. ■

A.2 Reduction from LSTSs to Petri Nets

Suppose we are given a LSTS T with finitely many propositional actions, and two configurations μ_0 and μ_1. Then there are only finitely many facts which can appear in any plan: those facts in the configurations μ_0 and μ_1 and those facts mentioned in one of the actions. Suppose there are k facts total, and for convenience let us rename them $F_1, \ldots, F_k$. Then we will associate to this LSTS a Petri net $N_T = (\widehat{P}, \widehat{T}, \phi)$ with places $\widehat{P} = \{p_1, \ldots, p_k\}$. Any configuration of the form $\mu = F_1^{\mu^1}, \ldots, F_k^{\mu^k}$ corresponds to a marking of the Petri net $\hat{\mu}$ such that $\hat{\mu}(p_i) = \mu^i$ for $1 \leq i \leq k$.

If T has actions $\{t_1, \ldots, t_s\}$ then the Petri net N_T will have transitions $\widehat{T} = \{\hat{t}_1, \ldots, \hat{t}_s\}$. If the action t_i is

$$t_i : F_1^{t^-_{i,1}}, \ldots, F_k^{t^-_{i,k}} \rightarrow F_1^{t^+_{i,1}}, \ldots, F_k^{t^+_{i,k}}$$

where $t^-_{i,j}, t^+_{i,j} \in \mathbb{N}$ for all $1 \leq j \leq k$, then ϕ satisfies

$$\phi(p_j, \hat{t}_i) = t^-_{i,j} \text{ and } \phi(\hat{t}_i, p_j) = t^+_{i,j} \text{ for all } 1 \leq j \leq k.$$

Then action t_i is enabled in configuration μ if and only if $t^-_{i,j} \leq \mu^j$ for all $1 \leq j \leq k$ if and only if transition $\hat{t}_i$ is enabled in marking $\hat{\mu}$. Note also that $\mu \rhd^1_{t_i} \mu'$ iff $\hat{\mu} \xrightarrow{\hat{t}_i} \hat{\mu}'$.

Notice that the size of the Petri net coverability problem is of the same order as the size of the LSTS reachability problem. Under an appropriate binary encoding the translation can be performed in polynomial time.

Proposition 6 *Under the above translation, $\mu_0 \rightsquigarrow^*_T \mu_1$ if and only if $\hat{\mu}_1 \in C(N_T, \hat{\mu}_0)$.*

Proof. We start with the forward direction. The proof is by induction on the length of the sequence of actions witnessing $\mu_0 \rightsquigarrow^*_T \mu_1$. The basis case of a length 0 sequence is trivial.
Induction Step:

Suppose that if $\mu_0 \rightsquigarrow^n_T \mu$ then $\hat{\mu} \in C(N_T, \hat{\mu}_0)$. Suppose also that $\mu_0 \rightsquigarrow^{n+1}_T \mu_1$. Then there is a configuration μ such that $\mu_0 \rightsquigarrow^n_T \mu \rightsquigarrow^1_{t_i} \mu_1$. By the inductive hypothesis $\hat{\mu} \in C(N_T, \hat{\mu}_0)$. Also, by definition, there is a configuration μ' such that $\mu \rhd^1_{t_i} \mu'$ where μ' contains μ_1. It is easy to check that $\hat{\mu} \xrightarrow{\hat{t}_i} \hat{\mu}'$ and $\hat{\mu}' \geq \hat{\mu}_1$. Thus $\hat{\mu}_1 \in C(N_T, \hat{\mu})$. By the transitivity of coverability we see that $\hat{\mu}_1 \in C(N_T, \hat{\mu}_0)$ as desired.

We now consider the other direction. The proof is by induction on the length of the firing sequence $\hat{\sigma}$ such that $\hat{\mu}_0 \xrightarrow{\hat{\sigma}} \hat{\mu}$ with $\hat{\mu} \geq \hat{\mu}_1$. The basis case of a length 0 firing sequence is trivial.
Induction Step:

Suppose that if $\hat{\mu}$ is coverable from $\hat{\mu}_0$ in n steps then $\mu_0 \rightsquigarrow^*_T \mu$. Suppose also that $\hat{\mu}_1$ is coverable from $\hat{\mu}_0$ in $n+1$ steps. Then there are markings $\hat{\mu}$ and $\hat{\mu}'$ such that $\hat{\mu}_0 \xrightarrow{\hat{\sigma}} \hat{\mu} \xrightarrow{\hat{t}_i} \hat{\mu}'$ with $\hat{\mu}' \geq \hat{\mu}_1$ and $\hat{\sigma}$ a firing sequence of length n. Then by the inductive hypothesis $\mu_0 \rightsquigarrow^*_T \mu$. It is also easy to see that since $\hat{\mu} \xrightarrow{\hat{t}_i} \hat{\mu}'$ then $\mu \rhd^1_{t_i} \mu'$, and since $\hat{\mu}' \geq \hat{\mu}_1$ we know that μ' contains μ_1. Thus $\mu \rightsquigarrow^*_T \mu_1$. By the transitivity of $\rightsquigarrow^*_T$ we conclude that $\mu_0 \rightsquigarrow^*_T \mu_1$ as desired. ■