Collaborative Planning With Confidentiality*

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

Paul Rowe
The MITRE Corporation
prowe@mitre.org

Andre Scedrov
University of Pennsylvania
scedrov@math.upenn.edu

January 25, 2011

Abstract

Collaboration among organizations or individuals is common. While these participants are often unwilling to share all their information with each other, some information sharing is unavoidable when achieving a common goal. The need to share information and the desire to keep it confidential are two competing notions which affect the outcome of a collaboration. This paper proposes a formal model of collaboration which addresses confidentiality concerns. We draw on the notion of a plan which originates in the AI literature. We use data confidentiality policies to assess confidentiality in transition systems whose actions have an equal number of predicates in their pre- and post-conditions. Under two natural notions of policy compliance, we show that it is PSPACE-complete to schedule a plan leading from a given initial state to a desired goal state while simultaneously deciding compliance with respect to the agents' policies.

1 Introduction

With information and resources becoming more distributed interaction with external services is becoming more important. There is a lot of software available which is designed to foster col-

*Research partially supported by OSD/ONR CIP/SW URI Software Quality and Infrastructure Protection for Diffuse Computing through ONR Grant N00014-01-1-0795, OSD/ONR CIP/SW URI Trustworthy Infrastructure, Mechanisms, and Experimentation for Diffuse Computing through ONR Grant N00014-04-1-0725, by ONR Grant N00014-07-1-1039, by OSD/AFOSR MURI Collaborative policies and assured information sharing, and by EPSRC Grant EP/D053625/1 Modularity and Resource Separation. Additional support from NSF Grants CNS-0429689, CNS-0524059, and CNS-0830949. This material is based upon work supported by the MURI program under AFOSR Grant No: FA9550-08-1-0352. Rowe's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author. Most of the work was done while Rowe was PhD student at the University of Pennsylvania. Approved for Public Release: 10-1569.

laboration through communication, resource sharing and data sharing. Collaborating agents are usually viewed to have a cooperative relationship in which they share a common goal. A common example can be found in companies which have a variety of departments working together to bring a product to market. In such situations confidentiality among departments is not always an obvious concern. However, just because parties are willing to collaborate does not mean they are ready to share all their information or resources. The distribution of health records is a good example.

Personal medical information can be quite sensitive. Hospitals take great care to ensure that information is shared and protected appropriately. There are quite detailed laws which specify exactly how a hospital may or may not distribute medical records. Some level of sharing is unavoidable. In order for a procedure to be covered by a patient's insurance company, the hospital must tell the insurance company what procedures were performed, as well as the diagnosis. Only then can the insurance company decide if it will cover the cost. But the insurance company should not be given other private information about the patient. Hospitals may also provide aggregate data on patients to students for educational purposes. This aggregate data should be appropriately sanitized for release to students. The same information, however, might not be acceptable for release to the general public.

When doing scientific research, researchers must find a balance between sharing too much information and not enough. On one hand, a principal goal of research is to obtain and publish results. On the other hand, if researchers provide their raw data to others too soon, another group may announce the results first. Data sharing is prominent in the fields of comparative genomics and systems biology where the focus is on integrating and analyzing large amounts of dynamic and growing biological data (see [**?**, **?**]). Scientific data sharing also has the converse problem. Scientists often rely on data from outside sources. The choice of whether or not to use outside data depends on the source of the data. The choice also depends on whether the outside data is consistent with the local confidential data. Research groups may have informal policies, or practices that determine when they are willing to incorporate outside data into their results.

Successful collaboration depends upon the proper balance between the protection and release of information or resources. In particular, collaboration is often not possible without some form of release of resources or information. The agents involved typically have some type of (possibly informal) confidentiality policy describing which types of data are allowed to be seen by which agents. The agents have an idea of which information releases are acceptable and which are not. Additionally, there may be external confidentiality policies imposed by law. Our focus is not on the conditions under which information is released, but rather on who is capable of seeing information which they should not see. The main confidentiality concern is that data might become available or visible to an agent who is prohibited from viewing it according to one of the policies.

In this paper we present a model of collaborative systems at an abstract level. We draw on previous work in planning and state transition systems and their connection to linear logic. We use local state transition systems in which we model local or confidential data through the use of a syntactic convention on predicate symbols. The global system state can then be thought of as

a collection of local states (comprised of confidential or High facts) together with some shared group state (comprised of group/public or Low facts). Indeed there are several High labels, with one for each agent and a single Low label. The confidential or local facts are those which are only visible and accessible to the local agent, although another agent may have another copy stored in their local state. We force the global state to change incrementally through local transitions that affect at most one local state at a time. A collaborative plan is then a sequence of transitions which takes the global state from some initial configuration to an agreed goal configuration. In fact, there may be many configurations that satisfy some goal condition.

We consider systems with well-balanced transitions which have the same number of facts in both the pre-conditions and the post-conditions. Intuitively, this means that each agent has a confidential (High) database, and there is a shared group (Low) database. The total number of fields in these databases is fixed, and the agents update the fields instead of creating new ones. Under this restriction, we find the complexity of scheduling a collaborative plan subject to certain confidentiality conditions. Our focus is on the interplay between confidentiality, or policy compliance, and goal reachability. Can the agents achieve their common goal while having some confidentiality guarantees?

We formulate two notions of compliance with confidentiality policies. The first notion, that of *system compliance*, requires that no execution of the system can violate any of the agents' confidentiality policies. The second notion, *plan compliance*, only requires a single execution, or plan, to be compliant with all the confidentiality policies. We show PSPACE-completeness of the following two problems which correspond respectively to the two confidentiality notions:

(I) Given a local state transition system, a set of confidentiality policies, an initial configuration, and a set of goal configurations, is the system compliant with the policies, and if so, schedule a plan leading from the initial configuration to one of the goal configurations (if one exists).

(II) Given a local state transition system, a set of confidentiality policies, an initial configuration, and a set of goal configurations, schedule a compliant plan leading from the initial configuration to one of the goal configurations (if one exists).

We also demonstrate the logical foundation of our approach in a variant of linear logic [**?**, **?**]. While this is not necessary in determining the complexity of the planning problem, it relates our approach to a number of similar formalisms which have had success in the past. On the one hand, there is a wide literature on viewing the classical planning problem in logical settings [**?**, **?**, **?**, **?**]. On the other hand, multiset rewriting formalisms have proved useful in the analysis of security protocols [**?**, **?**] as well as in the field of computation and concurrency [**?**, **?**, **?**, **?**, **?**, **?**]. This paper expands upon our preliminary report [**?**] with the addition of confidentiality policies and full proofs of the theorems.

The paper is structured as follows. Section 2 recalls single agent action systems and the classical planning problem. In section 3 we present state transition systems with multiple agents

and an extension which accounts for private data. We define collaborative planning in this setting. Section 4 discusses the relevant privacy properties. Logical foundation of our formalism is presented in section 5. Section 6 considers the complexity of the planning problem with privacy. We discuss related work in section 7. We present conclusions and future work in section 8.

2 Action Systems

A typical problem in Artificial Intelligence is that of a robot manipulating its environment in order to achieve some desired configuration. The robot is an agent which has limited abilities to sense and interact with its environment. The robot has at its disposal a set of actions which can change the state of the environment. The planning problem is that of trying to find a sequence of actions which will transform the environment from an initial state into a specified goal state.

These notions are formalized in *action systems* which specify how to describe the environment, what actions are available to the agent and how those actions affect the environment. The environment is characterized by a finite set of objects and relationships between those objects. Each action changes the relationships between the objects. Old relationships may be destroyed while new ones are simultaneously created. Each action is enabled by certain relationships between the objects. This means that not every action can be applied in every configuration of the environment.

To describe the environment we use a finite first order language without function symbols. Specifically, we use a finite set of constants to represent the objects of the environment. A finite set of predicates, will represent the possible relationships between the objects. A closed atomic formula will be called a *fact*. We define a *state* of the environment to be a set of facts. To illustrate the definitions we use an example common from the literature: the blocks world [**?**, **?**, **?**].

In this example there are three blocks represented by the constants a, b, and c. There are five predicates with the following interpretations:

ONTABLE(x) : x is on the table,
ON(x, y) : x is on top of y,
CLEAR(x) : nothing is on top of x,
HOLDS(x) : the robot holds x,
HANDEMPTY : the robot's hand is empty.

We can now represent the relative positions of the blocks on the table. One possible state of the environment is the following:

$$\{\text{ONTABLE}(a), \text{ON}(b, a), \text{CLEAR}(b), \text{ONTABLE}(c), \text{CLEAR}(c), \text{HANDEMPTY}\}.$$

This represents the situation in which b is stacked on a, and c sits on the table with nothing on top of it. We will typically use U, V, W and Z to represent states.

We now need to describe how the environment changes from one state to another. This is done through actions. Formally, an *action* is a map between states. Each action α is defined in terms of pre- and post-conditions denoted $pre(\alpha)$ and $post(\alpha)$ respectively. The domain of α is any state W in which $pre(\alpha)$ is a substate, (i.e. $pre(\alpha) \subseteq W$). If the state W is the domain of α, we say that W *enables* α. The result of applying the action is to replace $pre(\alpha)$ with $post(\alpha)$ in the state while leaving the rest of the state unchanged. We can thus represent the action as $pre(\alpha) \rightarrow post(\alpha)$. The agent can choose nondeterministically to apply any action which is enabled.

In our blocks world we have actions defined by the following conditions.

take(x):	$\{\text{HANDEMPTY}, \text{CLEAR}(x), \text{ONTABLE}(x)\} \rightarrow \{\text{HOLDS}(x)\}$
remove(x, y):	$\{\text{ON}(x, y), \text{HANDEMPTY}, \text{CLEAR}(x)\} \rightarrow \{\text{HOLDS}(x), \text{CLEAR}(y)\}$
stack(x, y):	$\{\text{HOLDS}(x), \text{CLEAR}(y)\} \rightarrow \{\text{HANDEMPTY}, \text{CLEAR}(x), \text{ON}(x, y)\}$
put(x):	$\{\text{HOLDS}(x)\} \rightarrow \{\text{ONTABLE}(x), \text{CLEAR}(x), \text{HANDEMPTY}\}$

The facts expressible in an action system determine a collection of possible states (*viz.*, the power set of the set of facts). We may have cause to view some states as inconsistent. For example, consider the state $\{\text{ON}(a, b), \text{ON}(b, a)\}$. Under our interpretation of the predicates, this state corresponds to blocks a and b being stacked on top of each other. Since this is physically impossible we want to introduce a mechanism that restricts the set of states we are willing to consider.

Let Σ be a subset of the states of an action system. These will be the states which we consider consistent. We say that an action system is compatible with Σ if every action preserves consistent states. Formally, for every state $W \in \Sigma$ and for every action α we have

if $pre(\alpha) \subseteq W$ then $\alpha(W) \in \Sigma$.

The planning problem is formulated in terms of an action system and set of states Σ which is compatible with the action system. Before we formulate the problem we must define a plan. A *plan* is a chain of actions. We say a plan *leads* from an initial state to a (complete) *goal* state if the following all hold.

(i) The first action of the plan is enabled by the initial state.

(ii) The resulting state of each action enables the next action in the plan.

(iii) The resulting state of the final action is the goal state.

Often a goal will not describe the entire environment. The goal could simply be any state in which block b is on block c. The position of a is irrelevant as long as it does not prohibit the goal situation. To express this idea we say that a plan leads from an initial state to a *partial goal* state

if (i) and (ii) hold, and in addition we have

(iii') The partial goal state is a substate of the state which results from the final action.

The planning problem is the following. Given an action system, an initial state W and partial goal state Z, does there exist a plan which leads from W to the partial goal Z?

The term *plan* is suggestive of the fact that the agent is meant to work out a plan before applying any actions. This is important because actions destroy old states. In many cases actions may be reversible, but this is not necessarily so. If an agent applies an irreversible action it may destroy its chances of reaching the goal. An abstract analysis before the agent starts performing actions can prevent such missteps.

3 State Transition Systems

3.1 Multiset Rewriting

Planning may also be considered to be part of a larger paradigm of state transition systems. State transition systems model concurrent computation by keeping track of a global state which is manipulated by multiple agents. Each agent uses a set of transitions in order to change the global state. By changing the state an agent may enable other agents to take further steps. We present state transition systems as multiset rewriting systems.

At the lowest level, we have a signature Σ of predicate symbols $P_1, P_2, \ldots$, and constant symbols $c_1, c_2, \ldots$. A *fact* is a ground, atomic predicate over multi-sorted terms. Facts have the form $P(\bar{t})$ where P is an n-ary predicate symbol and $\bar{t}$ is an n-tuple of terms, each with its own sort. A *state*, or *configuration* of the system is a finite multiset W of facts. The system evolves over time by way of a set of *transitions*. These transitions specify a *substate* which is to be rewritten as another substate. A transition will appear as $X \rightarrow X'$ where X and X' are multisets of facts. We call X the *enabling substate* and X' the *resulting substate* of the transition. We will use the convention that concatenation of multisets, such as WX, represents their multiset union. The transition $X \rightarrow X'$ thus transforms the state WX into the state WX'. It erases the multiset X and replaces it with X'.

We extend this notion of system evolution to that of reachability of a state Z from a state W. Given a set R of transition rules, if there is a sequence of (0 or more) transitions from R which transforms W into Z, then we say that Z is *reachable* from W using R.

Notice that these notions fit well with the classical ideas from planning. As discussed in [**?**, **?**], the environment state is now described by a multiset of facts instead of a set. Actions and transitions behave in the same fashion, creating new state information while destroying old information. In fact, from now on we will use the terms action and transition interchangeably. State reachability corresponds to the existence of a plan which transforms the environment from an initial situation to a goal situation.

3.2 Local State Transition Systems

We now want to extend these notions to a situation where each agent has access to private data which is inaccessible to all other agents. This requires us to extend the definitions a little. We now make a distinction between private facts and facts which are accessible to the whole group. For simplicity of exposition we restrict our terms to be constants and variables only (no function symbols). If we allow function symbols up to a fixed bounded depth, then the results of this paper continue to hold unchanged.

A signature Σ consists of predicate symbols with their arity, and many sorted constants and variables. As above, a fact is a ground, atomic predicate, but now we differentiate between private facts and group, or public facts using a syntactic convention. If a fact is private to participant A, we annotate the predicate with a subscript as, $P_A(\bar{t})$. We call this a private fact, and we will often say that agent A *owns* $P_A(\bar{t})$, or even A owns $\bar{t}$. Group facts are annotated with a prime marker as $P'(\bar{t})$. A participant is said to *know* a fact if it is a group fact or if that fact is owned by the participant. We extend this notation to multisets of facts, so the multiset X_A represents a multiset of facts all of which are owned by participant A. Similarly, X' represents a multiset consisting entirely of group facts. When denoting the union of X_A and X' we will write XX' whenever the agent A is clear from the context.

The distinction between confidential and public/group has the structure of a simple tree. There is one High label for each of the participants and there is a single Low label. One could easily imagine different structures to guide communication, such as having levels for certain subgroups of participants. In this work however, we only consider the simple structure illustrated in Figure 1.

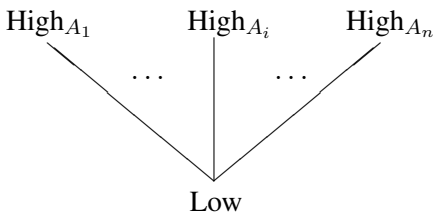


Figure 1: The simple tree structure of privacy.

A state, or configuration, of the system is now a multiset of group and private facts. Each agent thus only has partial knowledge of the global state. With this interpretation it is only natural that an agent can act based only on facts that it knows. For this reason, we restrict the enabling substate to contain private data owned by at most one agent. Similarly, the resulting substate will contain private data of at most one agent. Moreover, the agent who owns the private data in the enabling and resulting substates must be the same agent. We choose to denote a transition by $XX' \rightarrow_A YY'$, where each of X, X', Y, and Y' may be empty. The subscript A on the

arrow indicates that any private data which occurs belongs to A. When a transition contains facts owned by A we say it belongs to agent A. When it is clear from the context who owns a transition we will sometimes drop the subscript and write $XX' \to YY'$. We use R to denote a set of actions.

These transitions are local, in the sense that they only immediately depend on and affect data known locally by any one agent. Notice that the transitions themselves become private. For example, a transition of the form $X' \to_A Y'$ might represent a private algorithm applied to group data. Although other agents can see the input and output of the algorithm, they cannot perform the transformation on their own because they do not know the algorithm.

As before, an action $r : XX' \to_A YY'$ is applicable in a state W if $W = VXX'$ for some multiset of facts V. We say that the state W enables the transition r. The result of applying the action is a state $U = VYY'$. We use the notation $W \rhd_R U$ to mean that there is an action in R which can be applied to the state W to transform it into the state U. In particular, $W \rhd_r U$ means that the action r performs the transition. We let $\rhd_R^+$ and $\rhd_R^*$ denote the transitive closure and the reflexive, transitive closure of $\rhd_R$ respectively. Henceforth we will assume that each agent has actions which will copy group facts into private facts, such as

$$P'(\bar{t}) \to_A P'(\bar{t})P_A(\bar{t}).$$

Formally, a *local state transition system* T is a tuple (Σ, I, R_T), where Σ is a signature, I is a set of agents, and R_T is the set of (local) actions available to those agents.

Recall that, in the previous section, we had reason to consider plans with partial goals. Here again, we will develop a similar notion. We write $W \rightsquigarrow_{R_T} Z$ to mean that $W \rhd_{R_T} ZU$ for some multiset of facts U. For example with the action $r : XX' \to_A YY'$, we find that $WXX' \rightsquigarrow_r YY'$, since $WXX' \rhd_r WYY'$. We define $\rightsquigarrow_{R_T}^+$ and $\rightsquigarrow_{R_T}^*$ to be the transitive closure and the reflexive, transitive closure of $\rightsquigarrow_{R_T}$ respectively. We say that the partial configuration Z is reachable from state W with transition set R_T if $W \rightsquigarrow_{R_T}^* Z$.

We choose to visualize plans in this setting as non-branching trees (*i.e.*, directed chains of nodes) with labels on all edges and nodes.

Definition 3.1 *A* collaborative plan based on R_T *which* (exactly) leads *from an initial state W to a (complete) goal state Z is a labeled, non-branching tree whose labels satisfy the following.*

(i) *Edges are labeled with actions from R_T, and nodes are labeled with states.*

(ii) *The label of each node enables the label of its outgoing edge.*

(iii) *The label of the root is W.*

(iv) *The label of the leaf is Z.*

Thus to say that there exists a collaborative plan leading exactly from W to Z is the same as saying $W \rhd_{R_T}^* Z$. As with action systems, we extend this definition to partial goals as follows.

Definition 3.2 *A* collaborative plan based on R_T *which leads from an initial state* W *to a* partial goal Z *is a labeled, non-branching tree whose labels satisfy (i), (ii), and (iii) above, and which also satisfies*

(iv') *The label of the leaf is* ZU *for some multiset* U.

Here again, to say that there exists a collaborative plan leading from W to the partial goal Z, is equivalent to saying $W \rightsquigarrow^*_{R_T} Z$. The plan itself, though, actually encodes the sequence of actions used. In practice, a partial configuration may still be too specific, so we will focus on some set of partial goals $Z_1, \ldots Z_k$. We will then ask if there is a plan leading from W to *any* of these partial goals. That is, we ask if $W \rightsquigarrow^*_{R_T} Z_i$ for some $1 \leq i \leq k$.

In order to have a notation for the label of a node, we define, for each node w, its label with respect to an initial configuration W inductively as follows. For the root w_0,

$$\texttt{value}_{w_0}(W) := W.$$

For any edge (v, w), labeled by $XX' \rightarrow_A YY'$, if $\texttt{value}_v(W)$ is defined and $\texttt{value}_v(W) = VXX'$ for some V, then

$$\texttt{value}_w(W) := VYY'.$$

Otherwise it is undefined.

Example 3.1

We start with a secret sharing example. We will say that the triple $(A, 15, pky)$ means that "Alice's passkey is 15", and that the triple $(B, 7, share)$ mean that "Bob's share is 7". Similarly, $(A, 8, share)$ means that "Alice's share is 8". Then the initial configuration W of the system might look like the following:

$$W \quad = \quad P_A(A, 15, pky), S_A(B, 7, share), Q_B(B, 7, share)$$

That is, Alice knows both her passkey and Bob's share, while Bob only knows his share. If Charlie also begins with some information then we would need to explicitly add more facts to the configuration. Before continuing the example we must explain how the agents can transform the configuration. Let Alice's actions include

$$r_1 : P_A(A, 15, pky)S_A(B, 7, share) \rightarrow_A P_A(A, 15, pky)S_A(B, 7, share)P'(A, 8, share)$$

and let Bob's actions include

$$r_2 : Q_B(B, 7, share)P'(A, 8, share) \rightarrow_B Q_B(A, 15, pky)$$

When $R = \{r_1, r_2\}$, by applying r_1 and r_2 in that order, we find that

$$P_A(A, 15, pky)S_A(B, 7, share)Q_B(B, 7, share) \rightsquigarrow_R^* Q_B(A, 15, pky)$$

which means that Bob is able to learn Alice's passkey, 15. Notice that, in this example, the passkey never shows up in any public predicate. So if the above system evolution happens in the presence of a third agent, Charlie, then Charlie will not learn the passkey.

4 Data Confidentiality Policies

In this section we present the relevant definitions for expressing confidentiality violations in systems. We present data confidentiality policies and two types of policy violations. We also present two versions of the collaborative planning problem with confidentiality.

4.1 Motivation and Definitions

It is natural for the sensitivity of information to depend on both the context of the collaboration as well as the specific agents involved. For example, Alice may reveal information to a colleague when out socializing that she would not reveal to the same person while at work. Similarly, at work, Alice will likely be willing to reveal some information to certain people but not to others. Thus our formalism should be flexible enough to express these types of subtleties.

In formalizing what it means for Alice's secrets to leak, we first need to know where Alice is willing to let her information flow. We will assume that each agent has a data confidentiality policy which specifies which pieces of data other agents are prohibited from learning. In order to motivate the definitions which follow, we consider a scenario from the medical field.

Scenario: Medical Test.

Consider a patient at a hospital who needs a medical test performed in order for her doctor to provide a diagnosis and possibly prescribe appropriate medication. Such a task will involve not only the patient and her doctor. It will also involve a nurse to draw the blood, a lab technician to perform the test, and a secretary at the front desk to take care of the paperwork. Although there may be other agents typically involved in this task (such as an insurance company) we will limit ourselves to these agents for simplicity. Our focus in this scenario is on which pieces of information the patient is willing to share with the different agents, and in what combination.

In order for the test to be performed, the patient will request and schedule a test with the receptionist. The receptionist anonymizes the patient by giving her an ID number. The patient then goes to the nurse (with her ID number, not her name) to get a test sample taken. The nurse sends the test sample to a lab where the test is performed and the result is determined. The lab can then pass on the result to the doctor. We assume here that the doctor can recover the

Agent	Critical Information
Receptionist	(Name, Test Result), . . .
Nurse	(Name, ID ♯), . . .
Lab Technician	(Name, Test Result), . . .
Doctor	*No Restrictions*

Figure 2: Part of a patient's data confidentiality policy.

patient's name from the ID number. At this point the doctor can make a diagnosis and prescribe the appropriate medication.

This might be formalized by the following set of actions:

$$\begin{aligned}
P_{patient}(name, test_type) &\rightarrow_{patient} P'(name, test_type) \\
R_{reception}(name, test_type) &\rightarrow_{reception} R_{reception}(ID\#, test_type) \\
N_{nurse}(ID\#, test_type) &\rightarrow_{nurse} N_{nurse}(ID\#, test_sample) \\
L_{lab}(ID\#, test_sample) &\rightarrow_{lab} L_{lab}(ID\#, test_result) \\
D_{doctor}(ID\#, test_result) &\rightarrow_{doctor} D_{doctor}(name, diagnosis, prescription)
\end{aligned}$$

Of course, to allow the proper flow of information each agent would also need an action which publishes the appropriate information to a public predicate, as well as an action which reads information from a public predicate into their local predicate. We could then apply each of the above actions in order, interspersed with the appropriate read and publish actions to conclude that $P_{patient}(name, test\_type) \rightsquigarrow^* D_{doctor}(name, diagnosis, prescription)$.

We assume the patient has some idea of which information should not be learned by each of the agents. This information is specified in the patient's data confidentiality policy, a partial example of which can be found in Figure 2. For example, the secretary has no need to learn the patient's name in combination with the test result. Similarly, the point of anonymizing the patient is so that the nurse does not learn both the patient's name and ID number together. Although the lab technician must learn the result of the test, she should not know who the result pertains to. That is, she should not know both the name and the test result together, even though she can learn both the ID number and test result. In this scenario we assume the patient has no problem with the doctor learning any medical information.

Whether or not this policy is met will depend on a number of things, but in particular, it will depend on the actions the other agents have. If the lab technician can read $P'(name, test\_type)$ when the patient publishes it, then she may be able to combine it with $L_{lab}(ID\#, test\_result)$ to violate the patient's policy. However, if none of the lab technician's read actions have a pre-condition matching $P'(name, test\_type)$, then she cannot do this.

The reader should not be concerned with the specifics of this scenario, but rather with the expressivity of data policies. The patient's policy is some way of indicating which configurations

she considers as a violation of her confidential information. Although we only listed part of the patient's policy in Figure 2, each agent may have a similar policy. Indeed, there may be some external policy which is mandated by law.

In Figure 2 we simply used a tuple of values to specify data flows which are not allowed. In our formalism, these tuples actually represent a set of partial configurations which are considered violations of the confidentiality policy. For example, the patient's policy specifies that the combination of $(name, test\_result)$ should not be learned by the lab technician. The patient may consider that the lab technician knows these values if they show up in *any* of her private predicates or in a public predicate. Thus a configuration containing $L_{lab}(name)L_{lab}(test\_result)$ violates the policy just as $L_{lab}(name, test\_result)$ does. However, the patient may not mind the former configuration because the name and test results are not explicitly linked.

If a configuration violates the confidentiality policy of one of the agents, then we call it a *critical configuration*. A confidentiality policy is just a way of expressing which configurations the agent considers to be critical. Notice that since policies only specify which configurations must *not* occur, no conflict arises when combining policies. The combination of two policies simply specifies the union of their critical configurations.

When modeling real situations these policies serve as constraints in the model, which may not represent real constraints in the actual system. For example a policy which is mandated by law will impose real restrictions on the system, while a patient's policy may simply represent her personal preferences. With enough knowledge of the system, she can use this policy to guide decisions about what collaborations she enters into.

Our only assumption on confidentiality policies is that they specify which configurations are critical (and implicity which are not). Discussions of how these policies may be represented is beyond the scope of this work. It would be an interesting area for further research to join our formalism with one of the numerous privacy languages already in existence [**?**, **?**]. We note, though, that our complexity results are not affected by the details of how policies are represented, as long they satisfy certain assumptions to be detailed later. We henceforth assume that there is some fixed representation of these policies.

Equipped with the notion of critical configurations we are now ready to formally define what it means both for a system to comply with the policies and likewise for a plan to comply with the policies.

Definition 4.1 *Given a set of confidentiality policies, a local state transition system in initial configuration W is said to be* compliant *if every critical configuration defined by the policies is unreachable from W.*

Definition 4.2 *Given a set of policies, a plan is said to be* compliant *if none of the configurations in the plan are critical.*

So a compliant plan is one which avoids the critical configurations. It should be clear that

a system is compliant if and only if every plan in the system is compliant. However, there are many systems which have both compliant and non-compliant plans.

These definitions differ somewhat from our original privacy/secrecy condition in the preliminary version of this paper [**?**]. We used the following definition instead.

Definition 4.3 *We say that a local state transition system in initial configuration W,* protects the privacy of agent A *if every term t which, in the initial configuration W, occurs only in private (High) predicates of A, also occurs only in private (High) predicates of A in any reachable configuration.*

In fact, we can express this condition as a data confidentiality policy which is entirely determined by the initial configuration. The critical configurations of this policy are those configurations which contain predicates of the form $P_B(t, \bar{u})$, where t is a term which is to be protected, B is an agent not equal to A, and $\bar{u}$ is an arbitrary (and possibly empty) tuple of terms. Similarly, configurations containing $P'(t, \bar{u})$ would be critical. In particular, this policy does not distinguish between other agents. We assumed that agent A was unwilling to reveal any secret term t to any agent.

In the end we considered this condition to be too restrictive, so we generalized the notion to data confidentiality policies which are much more flexible. This is also what caused us to consider the weaker notion of plan compliance. Even somewhat liberal policies may be too strong to satisfy system compliance. In Section 6 we discuss the effect these changes had on our complexity results and show that Corollary 6.6 in [**?**] is actually a corollary of Theorem 6.3 in this current paper.

4.2 The Collaborative Planning Problem with Confidentiality

While the classical planning problem asks only for a plan which leads from the initial configuration to a goal configuration, our setting requires more. We must also ask if the confidentiality policies of all the agents are respected. There seem to be two natural ways to do that which correspond to Definitions 4.1 and 4.2 respectively.

The Collaborative Planning Problem with System Compliance:

Given a local state transition system T, initial configuration W, a (partial) goal configuration Z, and a set of confidentiality policies, does the system comply with the policies, and does there exist a plan leading from W to Z?

The Collaborative Planning Problem with Plan Compliance:

Given a local state transition system T, initial configuration W, a (partial) goal configuration Z, and a set of confidentiality policies, is there a compliant plan which leads from W to Z?

A positive answer to the first version gives very strong guarantees about policy compliance. Each agent can be sure that even if the other agents deviate from the plan which exists, their

confidentiality policy will not be violated. The policies are respected regardless of the behavior of the agents. A positive answer to the second version is not nearly as strong. It simply says that policy compliance and goal reachability are not contradictory in the given system. It would still be possible for an agent to perform extra local computations after reaching the goal in order to learn secrets and violate another agent's policy. However, depending on the level of trust among the agents, this may still be an important question, especially if the answer to the first version is "no".

This could be a hard question to answer in the general case. In particular, if we put no restrictions on the form of the actions, then we could grow the global state to an arbitrarily large size. The general case for goal reachability (where there are no confidentiality policies) has the same complexity as the coverability problem for Petri nets [**?**, **?**, **?**, **?**]. Although this is decidable [**?**], it overestimates the complexity of the typical case.

It is possible to model a large class of collaborations using a fixed amount of total resources. We may assume that the total number of facts is fixed at all times. This is enforced by limiting the transitions to be well-balanced [**?**]. A *well-balanced* transition is a transition which has the same number of facts in the pre-condition and the post-condition (counting repetitions). In well-balanced local state transition systems, the total number of facts present in the global state remains constant.

Intuitively, this restriction forces each agent to have a buffer or database of a fixed size before the collaboration. The agents may update values in this buffer and erase values to leave empty slots, but they may not change the size of the buffer. This can be done in a natural way. Consider each predicate in the language as a field in some database. Then $P_A(t)$ can be interpreted as saying that the field P_A of A's private database is occupied by t. Although using well-balanced actions forces us to fix the number of fields of this database, there is no *a priori* bound on the number of fields we may choose.

In reality, the model is more flexible than that. There is one global buffer of a fixed size, and the agents are free to release some fields for use by the group and claim others from the group. The model allows for a potential fight for space resources which could result in a form of denial of service. Since we are considering situations in which the agents are mostly cooperative and since our security focus is on the inappropriate release of information, we do not explore this dynamic of well-balanced systems.

While at first the well-balanced condition may seem like a big restriction, in practice we are still able to model most scenarios in a natural way. To demonstrate this, we transform Example 3.1, which is not well-balanced, into a well-balanced example. Assume that our language has a special constant $*$. A predicate of the form $P_A(*)$ can be interpreted as saying that the field P_A is empty. We assume that confidentiality policies never refer to this constant. Alice's action may then be rewritten in the following well-balanced form.

$$r_1 : P_A(A, 15, pky)S_A(B, 7, share)P'(*) \rightarrow_A \\ P_A(A, 15, pky)S_A(B, 7, share)P'(A, 8, share)$$

Likewise, Bob's action can be rewritten in a well-balanced form.

$r_2 : Q_B(B, 7, share)P'(A, 8, share) \rightarrow_B Q_B(A, 15, pky)P'(*)$

In this way, if the initial configuration has m predicates (many of which may be of the form $P(*)$) we can model any non-balanced computation which does not grow the global state to a size greater than m. In particular, using well-balanced actions does not change the reachability of states, as long they are reachable in a non-balanced system evolution which limits the configuration size to m.

In Section 6 we will also refer to a slightly more general case of *non-lengthening* local state transition systems, for which in each action the number of facts in the post-condition is *at most* the number of facts in the pre-condition.

4.3 Remarks

In modeling a real world collaboration the agents may have a given protocol in mind which they would like to follow. The actions of each agent would then include the actions which model the agents behavior in the protocol. The actions and initial configuration then determine a space of collaborative plans or protocols which, by design, contains the protocol the agents had in mind. For a correctly designed protocol, the goal reachability part of the Collaborative Planning Problem with System Compliance should be trivial to show. We can just look at the sequence of actions which model a correct execution of the protocol. The hard part is proving that the system is in fact compliant with the confidentiality policies.

For a given agent, all the other agents can be viewed as a kind of adversary. Indeed the problem is akin to proving properties of security protocols in the presence of a Dolev-Yao attacker [**?**]. One key difference is that the nondeterminism is now spread out among all the agents, but these agents have bounded memory (in the case of well-balanced actions). The agents' also each have a potentially different set of cryptographic operations encoded into their actions, while the Dolev-Yao adversary has one fixed set of actions at its disposal. Another difference is that we do not accommodate agents who create fresh data. This allows us to show that the collaborative planning problems are both decidable. In fact, in other work [**?**] we lift the well-balanced restriction and show that the problem with system compliance is still decidable, but the version with plan compliance becomes undecidable.

Because of the stateful nature of our formalism, we are able to ask privacy/secrecy questions which have not been studied extensively in the literature. Namely, can the other agents learn Alice's *current* confidential information? To illustrate this point let us return, once again, to the Example 3.1 above. Alice's action may not only release enough information to reveal her passkey, it may actually change her passkey in the process. The action may be changed to the following:

$$r_1 : P_A(A, 15, pky)S_A(B, 7, share)P'(*) \rightarrow_A \\ P_A(A, 21, pky)S_A(B, 7, share)P'(A, 8, share)$$

Notice that Alice's passkey has changed from 15 to 21.

Bob would still be able to learn Alice's *old* passkey, but he has no access to her *current* passkey. Her old passkey is obsolete and will not help Bob learn any sensitive secrets. The real danger arises when Bob is in possession of Alice's current private passkey. This may provide him access to more sensitive data. Our confidentiality policies may be expressive enough to specify this type of distinction between current and obsolete. However, for the moment we simply remark that a stateful approach is amenable to this distinction and we leave a more complete investigation of it for future work.

5 Foundation in Logic

In this section we present the logical foundation of our approach. We demonstrate a connection to a variation of linear logic known as affine logic. We include it for several reasons. First, it represents an important stage in the development of our ideas. It provides a kind of "sanity check" showing that we can formally ground our ideas in a well-understood formalism. Second, we find it interesting that the proper connection is to affine logic and not to linear logic. This is an important difference from MSR [**?**] which has been shown to have correspondences with linear logic [**?**]. We actually use this connection to affine logic rather explicitly in other work [**?**, **?**] which helps to illuminate some of the differences from MSR and its connection to linear logic. Finally, it can be desirable to have such a logical foundation because we are often able to gain new insight by thinking in terms of a well established formalism. For example we may be able to apply existing tools, which work with some logical formalism, to the problem at hand.

Linear logic (LL), which was introduced in [**?**], is a resource-sensitive refinement of traditional logic. It is presented as a Gentzen style sequent calculus. A sequent of the form $\Gamma \vdash \Delta$ says roughly that the resources in the multiset Γ can be used to produce the multiset Δ. Linear logic differs in a number of ways from traditional logic, but most notably it does not allow the rules of weakening or contraction.

The rule of contraction is given by

$$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}$$

It says that if we can produce Δ using Γ and two copies of A, then we can also produce Δ with Γ and one copy of A. Weakening is the opposite rule and is given by

$$\frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}$$

It says that if we can produce Δ using Γ, then we can also produce Δ with Γ and A, for any A.

The effect of disallowing the contraction rule is that it forces us to use each resource *at most* once. The effect of disallowing the weakening rule is that it forces us to use each resource *at least* once. This is why linear logic is popular when trying to model resource-sensitive systems. It allows the available resources to grow and shrink, in contrast to traditional logic in which the resources grow monotonically. Linear logic also provides a mechanism which allows certain resources to be used any number of times. This will be useful when we use logic to model actions.

At a lower level, linear logic splits the traditional connectives defining conjunction and disjunction into two forms. For example, the traditional conjunction $\wedge$ is split into $\otimes$ (multiplicative conjunction) and $\&$ (additive conjunction). A derivation of a $\otimes$ conjunction forbids any sharing of the resources used to derive each conjunct. In contrast, a derivation of a $\&$ conjunction requires all resources to be shared.

Affine logic (AL) is the variation of linear logic which allows weakening, but still disallows contraction. Thus, in affine logic, we may use each resource either once or not at all. Affine logic is therefore an appropriate logic to use when we want to model the reachability of *partial* goals. It allows us to work with the relevant resources in arbitrary contexts.

There is a well-known correspondence between state transition systems and linear logic. We extend this connection to one between local state transition systems and affine logic. By encoding the objects of local state transition systems as logical formulas we are able to relate partial goal reachability with derivability of certain Gentzen sequents within affine logic.

Our translation of local state transition systems into affine logic follows closely the translation of ordinary state transition systems into linear logic. There is one notable difference which deserves mention here. If we want our translation to preserve all key elements of local state transition systems, we must preserve the information which allows us to determine which action is allowed to be used by which participant. For this reason we translate actions into linear implications with a special constant which acts as a "lock", as described below.

For a local state transition system T, multisets of facts are encoded using the multiplicative conjunction, $\otimes$ (which is commutative).

$$\ulcorner S \urcorner = \bigotimes S.$$

The empty multiset is encoded as the multiplicative unit.

$$\ulcorner . \urcorner = \mathbf{1}.$$

Transition rules are encoded as linear implication. This is where we use the "lock". For each agent A we have a constant symbol q_A. We then encode transition rules using linear implication as follows.

$$\ulcorner XX' \rightarrow_A YY' \urcorner = q_A \otimes \ulcorner XX' \urcorner \multimap q_A \otimes \ulcorner YY' \urcorner$$

This technique forces derivations in affine logic to contain all the relevant information about which agent is applying an action at a given point. Now, for the rule set R_T we can define

$$\ulcorner R_T \urcorner = \{\ulcorner r \urcorner : r \in R_T\}.$$

Since locks of the form q_A are part of the linear implications we must also include these constants as part of the global configuration. If I is the set of participants of T, then we define

$$\ulcorner I \urcorner = \bigotimes \{q_A : A \in I\}.$$

We use a notion of provability which is based on an intuitionistic version of affine logic in which the sequents have the form

$$\Gamma; \Delta \vdash C$$

where Δ is a linear context, Γ is an unrestricted context in which the resources may be used as many times as necessary, and C is a single formula (see [**?**]). The relevant rules of affine logic are presented in the appendix. Under our encoding of local state transition systems we are able to achieve the following results.

Theorem 5.1 (Soundness) *For every pair of states W, Z, and every rule set R_T defining a participant set I, if $W \rightsquigarrow^*_{R_T} Z$ then $\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash \ulcorner I \urcorner \otimes \ulcorner Z \urcorner$ is derivable in affine logic.*

Theorem 5.2 (Completeness) *For every pair of states W, Z and every rule set R_T defining a participant set I, if $\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash \ulcorner I \urcorner \otimes \ulcorner Z \urcorner$ is derivable in affine logic then $W \rightsquigarrow^*_{R_T} Z$.*

In order to prove these results we will pass through soundness and completeness results which relate exact reachability to derivability in linear logic (Lemmas 5.1 and 5.2 respectively). We then prove Lemma 5.3 which relates derivability of sequents in linear logic to derivability of sequents in affine logic. This last lemma allows us to lift the soundness and completeness results with respect to linear logic to the corresponding results with respect to affine logic. Since we are concerned with sequents in both linear logic and affine logic we will write $\Gamma; \Delta \vdash_{LL} C$ for sequents which are meant to be derived in linear logic, and we write $\Gamma; \Delta \vdash_{AL} C$ for sequents which are meant to be derived in affine logic.

Lemma 5.1 *For every pair of states W, Z and every rule set R_T defining a participant set I, if $W \rhd^*_{R_T} Z$ then $\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner$ is derivable.*

Proof: (by induction on the length of the action sequence) We consider two basis cases:

Length 0 transition sequence: $W \rhd^0_{R_T} Z$.

That means that $W = Z$ and $\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner W \urcorner$ is the conclusion of the identity rule.

Length 1 transition sequence: $W \rhd^1_{R_T} Z$.

So Z is reachable from W using only one rule from R_T. Label that rule by $r : X_\alpha X' \to_\alpha Y_\alpha Y'$. We get the following derivation:

$$\dfrac{\dfrac{\dfrac{\ulcorner R_T \urcorner; q_\alpha \otimes \ulcorner X_\alpha \urcorner \otimes \ulcorner X' \urcorner \vdash_{LL} q_\alpha \otimes \ulcorner X_\alpha \urcorner \otimes \ulcorner X' \urcorner \qquad \ulcorner R_T \urcorner; q_\alpha \otimes \ulcorner Y_\alpha \urcorner \otimes \ulcorner Y' \urcorner, V \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner}{\ulcorner R_T \urcorner; q_\alpha \otimes \ulcorner X_\alpha \urcorner \otimes \ulcorner X' \urcorner \multimap q_\alpha \otimes \ulcorner Y_\alpha \urcorner \otimes \ulcorner Y' \urcorner, q_\alpha \otimes \ulcorner X_\alpha \urcorner \otimes \ulcorner X' \urcorner, V \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner} \multimap \ell}{\ulcorner R_T \urcorner; q_\alpha \otimes \ulcorner X_\alpha \urcorner \otimes \ulcorner X' \urcorner \multimap q_\alpha \otimes \ulcorner Y_\alpha \urcorner \otimes \ulcorner Y' \urcorner, \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner} \otimes\ell}{\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner} \text{ clone}$$

Reading this derivation from the bottom up, we first clone the action from the unrestricted context into the linear context. We then split the state into the precondition of the action, $q_\alpha \otimes \ulcorner X_\alpha \urcorner \otimes \ulcorner X' \urcorner$ and the rest of the state $V = (\ulcorner I \urcorner \otimes \ulcorner W \urcorner) \setminus (q_\alpha \otimes \ulcorner X_\alpha \urcorner \otimes \ulcorner X' \urcorner)$. We then apply the $\multimap \ell$ rule to find two sequents. The left sequent is obtained by the identity rule. The right sequent can be separated by the use of $\otimes r$ to reduce it to two applications of the identity. This shows that the lemma holds for length 1 transition sequences.

Induction Step:

Assume it holds for transition sequences of length $\leq n$. Now assume that $W \rhd^{n+1}_{R_T} Z$. Then we know that there is some U such that $W \rhd^n_{R_T} U$ and $U \rhd^1_{R_T} Z$. Therefore our induction hypothesis tells us that we have two derivations ending in $\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner U \urcorner$ and in $\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner U \urcorner \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner$. We can apply the cut rule with these two conclusions to obtain a derivation with the conclusion $\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner$ as desired. ■

Lemma 5.2 *For every pair of states W, Z and every rule set R_T defining a participant set I, if $\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner$ is derivable then $W \rhd^*_{R_T} Z$.*

Proof: Completeness is much more complicated than soundness. The reason for this is that there may be many different LL-derivations of a given formula. We cannot simply assume that the derivation which is assumed to exist has a nice form that corresponds to the form of a derivation which arises in the proof of soundness. We will briefly sketch the ideas here noting that the key to this argument is the ability to permute certain rules above or below others. We direct the reader to [**?**] for the proofs of these permutation rules. We also note that LL completeness has been shown for rewriting systems before in [**?**, **?**].

By cut elimination there is a cut-free derivation of the sequent

$$\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner.$$

In particular, this derivation satisfies the sub-formula property which states that any formula occurring in the derivation must be a sub-formula of the formulas occurring in the conclusion. This allows us to restrict our attention to derivations with only the following rules: id, clone, $\otimes\ell$, $\otimes r$, $\mathbf{1}\ell$, $\mathbf{1}r$, $\multimap \ell$ and $\multimap r$. Furthermore, we can rule out derivations which use $\multimap r$ since we

do not use nested implications. As mentioned above, these rules may be permuted in restricted ways as proven in [**?**]. These permutation results allow us to assume that clone appears below all other rules, $\otimes r$ occurs just below id or $\mathbf{1}r$ which occur at or near the leaves, and that $\otimes\ell$ and $\mathbf{1}\ell$ are applied greedily (when reading the derivation from bottom to top). We must respect the nesting structure, so that in decomposing $(A \otimes B) \multimap C$ we must apply the $\multimap \ell$ rule below the $\otimes\ell$ rule. The derivation can thus be assumed to have the form seen in Figure 3.

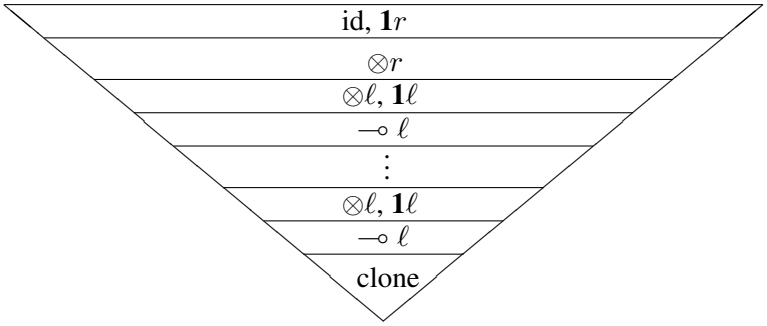


Figure 3: The form of a permuted proof.

Note that the conclusion of each $\multimap \ell$ is of the form $\ulcorner R_T \urcorner; \ulcorner \bar{r_i} \urcorner, \ulcorner I \urcorner \otimes \ulcorner W_i \urcorner \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner$, where $\ulcorner \bar{r_i} \urcorner$ is a multiset of m - i encodings of actions which have yet to be decomposed, and where there are m applications of $\multimap \ell$ in the derivation. In the conclusion of the bottom application of $\multimap \ell$ is $W_0 = W$. In the right premise of the top application of $\multimap \ell$ is $W_m = Z$.

Claim: There is a transition sequence $r_1, \ldots, r_m$ such that

$$W = W_0 \rhd_{r_1} W_1 \rhd_{r_2} \cdots \rhd_{r_m} W_m = Z$$

allowing us to conclude $W \rhd^*_{R_T} Z$.

The claim follows from the simple observation that if the occurrence of $\multimap \ell$ between W_i and W_{i+1} decomposes $q_\alpha \otimes \ulcorner X_\alpha \urcorner \otimes \ulcorner X' \urcorner \multimap q_\alpha \otimes \ulcorner Y_\alpha \urcorner \otimes \ulcorner Y' \urcorner$ then $W_i = X_\alpha X' V$ for some V and $W_{i+1} = Y_\alpha Y' V$ for the same V. Thus $W_i \rhd_{R_T} W_{i+1}$ via $X_\alpha X' \rightarrow_\alpha Y_\alpha Y'$. This observation is easily turned into an inductive proof. ■

Lemma 5.3 *$\Gamma; W \vdash_{AL} Z$ is derivable if and only if there is some U such that $\Gamma; W \vdash_{LL} Z \otimes U$ is derivable.*

Proof: This lemma is an immediate corollary to Lemma 3.1 in [**?**]. The *if*-direction is immediate since every LL-derivation is also an AL-derivation. The *only if*-direction requires a simple induction on the structure of the LL-derivation. We direct the reader to [**?**] for more details. ■

Proof: *(of Theorems 5.1 and 5.2)* The three lemmas combine in the following way to imply soundness and completeness with respect to AL.

$$W \rightsquigarrow^*_{R_T} Z \Leftrightarrow \text{(by definition)}$$
$$\exists U \text{ s.t. } W \rhd^*_{R_T} ZU \Leftrightarrow \text{(Lemmas 5.1, 5.2)}$$
$$\exists U \text{ s.t. } \ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash_{LL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner \otimes \ulcorner U \urcorner \Leftrightarrow \text{(Lemma 5.3)}$$
$$\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash_{AL} \ulcorner I \urcorner \otimes \ulcorner Z \urcorner$$

■

The above theorems relate the existence of a plan to derivability in pure affine logic. It is also possible to translate local state transition systems into affine logic *theories*. Namely, for every action of the form $X \rightarrow_A Y$ we include a new axiom of the form $q_A \otimes X \vdash q_A \otimes Y$. The set of axioms of a theory T is denoted by $\mathtt{Ax}_T$. Theories with axioms of this form are called (pure) *Horn theories*.

It can be easily seen that finite Horn theories can be faithfully represented in the formalism above, by listing the non-logical axioms, $\mathtt{Ax_T}$, in the unrestricted context, Γ.

Under this interpretation, the existence of a plan leading from an initial state W to an exact goal Z can be shown to be equivalent to derivability of the sequent $\ulcorner I \urcorner \otimes W \vdash Z$ in the corresponding Horn LL-theory. Also, $\ulcorner I \urcorner \otimes W \vdash Z$ is derivable in the corresponding AL-theory if and only if $\ulcorner I \urcorner \otimes W \vdash Z \otimes U$ is derivable in the LL-theory for some U [**?**].

The definition of system compliance is essentially an unreachability condition. Namely, we expect that every critical configuration should be unreachable. Under our translation into affine logic, this is the same as saying that the corresponding sequents should not be deducible. Translating the definition of plan compliance is more subtle. It corresponds to a certain kind of normal form proof not containing sequents whose left hand side represents a critical configuration. We do not investigate this relation further.

6 Complexity

In this section we determine exactly the complexity class of both the Collaborative Planning Problem with System Compliance and the Collaborative Planning Problem with Plan Compliance with well-balanced transitions. In Section 6.1 we demonstrate a PSPACE lower bound for these problems. Section 6.2 demonstrates a PSPACE upper bound for both of these problems.

6.1 Lower PSPACE Bounds

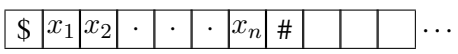
We start with a reduction from the PSPACE-complete problem IN-PLACE ACCEPTANCE for non-deterministic Turing machines [**?**].

Theorem 6.1 *Any non-deterministic Turing machine M that accepts in space n can be faithfully represented within well-balanced local state transition systems.*

Proof. Let M be a non-deterministic machine that accepts in space n.
Without loss of generality, we assume the following:

(a) M has only one tape, which is one-way infinite to the right. The leftmost cell (numbered by 0) contains the marker \$ unerased.

(b) Initially, an *input* string, say $x_1x_2\ldots x_n$, is written in cells 1, 2,. . . ,n on the tape. In addition, a special marker # is written in the $(n+1)$-th cell.



(c) The program of M contains no instruction that could erase either \$ or #. There is no instruction that could move the head of M either to the right when M scans symbol #, or to the left when M scans symbol \$. As a result, M acts in the space between the two unerased markers.

(d) Finally, M has only one *accepting* state, and, moreover, all *accepting* configurations in space n are of one and the same form.

For each n, we design a local state transition system T_n (with a single agent A) as follows. We introduce the following 0-ary predicates (propositions), which are assumed to be private predicates of A:

(a) $R_{i,\xi}$ stands for *"the i-th cell contains symbol ξ"*,
here $i=0,1,\ldots,n+1$, ξ is a symbol of the tape alphabet of M,

(b) $S_{j,q}$ means *"the j-th cell is scanned by M in state q"*,
here $j=0,1,\ldots,n+1$, q is a state of M.

Given an *instantaneous description* (*configuration*) of M in space n - that M scans j-th cell in state q, when a string $\xi_0\xi_1\xi_2\ldots\xi_i\ldots\xi_n\xi_{n+1}$ is written left-justified on the otherwise blank tape, we will represent it by a configuration of T_n of the form

$$S_{j,q}R_{0,\xi_0}R_{1,\xi_1}R_{2,\xi_2}\cdots R_{n,\xi_n}R_{n+1,\xi_{n+1}}. \tag{1}$$

Necessarily, $\xi_0 = \$$, and $\xi_{n+1} = \#$.
Each of the instructions of M, $q\xi\to q'\eta D$:

> *"if in state q looking at symbol ξ, replace it by η, move the tape head one cell in direction D along the tape, and go into state q'"*,

is specified by the set of $n+2$ actions of the form, $i=0,1,\ldots,n+1$:

$$S_{i,q}R_{i,\xi} \to_A S_{i_D,q'}R_{i,\eta}, \tag{2}$$

where $i_D := i+1$ if D is *right*, and $i_D := i-1$ if D is *left*, and $i_D := i$, otherwise.
(Neither $i_{right} = n+2$, nor $i_{left} = -1$ can occur.)
We denote the set of all these actions by R_{T_n}.

Lemma 6.1 *Given an input string* $x_1x_2\ldots x_n$ *of length* n*, let* W_n *be a configuration of* T_n *of the form (1), which represents the* initial *configuration of* M *with the input string* $x_1x_2\ldots x_n$*, and let* Z_n *be a configuration of* T_n *of the form (1), which represents the unique* accepting *configuration of* M *of length* n*.*
The following propositions are pairwise equivalent:

(a) *The input string* $x_1x_2\ldots x_n$ *is accepted by* M*.*

(b) $W_n \rhd^*_{R_{T_n}} Z_n$

(c) $W_n \rightsquigarrow^*_{R_{T_n}} Z_n$

(d) *There exists a finite plan based on* R_{T_n} *that (exactly) leads from* W_n *to* Z_n*.*

Proof. Our encoding proceeds in a straight course, so that given any successful non-deterministic computation in space n that leads from the initial configuration represented by W_n to the accepting configuration represented by Z_n we can rewrite it as a sequence of actions from R_{T_n} leading from W_n to Z_n. This is done simply by writing (in order) the actions from R_{T_n} which correspond to the transitions used in the successful computation. This shows that $W_n \rhd^*_{R_{T_n}} Z_n$. Hence (a) implies (b). Item (c) follows from (b) by the definitions, and we noted earlier that (d) and (b) are equivalent.

The most complicated direction is that such a straightforward encoding is *faithful*. Suppose that $W_n \rightsquigarrow^*_{R_{T_n}} Z_n$. Then, for some V_1 we have $W_n \rhd^*_{R_{T_n}} Z_nV_1$. That is, there is a finite plan $\mathcal{P}$ based on R_{T_n} that exactly leads from W_n to Z_nV_1.

Since each of the actions (2) is *well-balanced*, and the number of facts in both W_n and Z_n is just the same $n+3$, this V_1 must be empty, which means that for the leaf ℓ in $\mathcal{P}$, the following holds:

$$\texttt{value}_\ell(W_n) = Z_n.$$

Hence $W_n \rhd^*_{R_{T_n}} Z_n$, and so (c) implies (b).

On the other hand, because of the specific form of our actions (2) any $\texttt{value}_v(W_n)$ in $\mathcal{P}$ is of the form (1), and, hence, represents a configuration of M in space n.

Passing through this $\mathcal{P}$ from its leaf to its root v_0, we prove that whatever vertex v we take, there is a successful non-deterministic computation performed by M leading from the configuration represented by $\texttt{value}_v(W_n)$ to the *accepting* configuration represented by Z_n.

In particular, since $\texttt{value}_{v_0}(W_n) = W_n$, we can conclude that the given input string $x_1x_2\ldots x_n$ is accepted by M, and so (b) implies (a). ■

Notice that the signature of T_n consists only of $O(n)$ 0-ary predicate symbols. The number of the constants invoked is zero. If we confine ourselves both to a fixed number of predicate symbols and to a fixed number of constants, Theorem 6.1 fails because of the *polytime* upper bound of Theorem 6.5. Nevertheless, the next Theorem show that we get PSPACE-hardness also at the opposite end of the "scale" *with a fixed number of predicate symbols and an unbounded number of constants*.

Theorem 6.2 *Any non-deterministic Turing machine M that accepts in space n can be faithfully represented in well-balanced local state transition systems with a fixed number of unary predicates.*

Proof. For each n, we will design a local state transition system T'_n (with a single agent A) by modifying the construction of Theorem 6.1 as follows.
Assume a list of constants:

$$c_0, c_1, c_2, \ldots, c_n, c_{n+1}.$$

We introduce the following unary predicate symbols, which are assumed to be private predicates of A:

(a) $\widehat{R}_\xi$, for every ξ from the tape alphabet of M.

The intended meaning of $\widehat{R}_\xi$ is:

$$\widehat{R}_\xi(c_i) = R_{i,\xi}.$$

(b) $\widehat{S}_q$, for every q from the set of states of M.

The intended meaning of $\widehat{S}_q$ is:

$$\widehat{S}_q(c_j) = S_{j,q}.$$

Given an *instantaneous description* (*configuration*) of M in space n - that M scans j-th cell in state q, when a string $\xi_0\xi_1\xi_2\ldots\xi_i\ldots\xi_n\xi_{n+1}$ is written left-justified on the otherwise blank tape, now it is represented as (cf. (1)):

$$\widehat{S}_q(c_j)\widehat{R}_{\xi_0}(c_0)\widehat{R}_{\xi_1}(c_1)\cdots\widehat{R}_{\xi_n}(c_n)\widehat{R}_{\xi_{n+1}}(c_{n+1}). \tag{3}$$

Each of the actions of the form (2) is rewritten as:

$$\widehat{S}_q(c_i)\widehat{R}_\xi(c_i) \rightarrow_A \widehat{S}_{q'}(c_{i_D})\widehat{R}_\eta(c_i). \tag{4}$$

The *faithfulness* of the modified encoding is established by

Lemma 6.2 *Given an input string $x_1x_2\ldots x_n$ of length n, let W_n be a configuration of T'_n of the form (3) that represents the* initial *configuration of M with the input string $x_1x_2\ldots x_n$, and let Z_n be a configuration of T'_n of the form (3) that represents the unique* accepting *configuration of M of length n.*
The following propositions are pairwise equivalent:

(a) *The input string $x_1x_2\ldots x_n$ is accepted by M.*

(b) $W_n \rhd^*_{R_{T'_n}} Z_n$

(c) $W_n \rightsquigarrow^*_{R_{T'_n}} Z_n$

(d) *There exists a finite plan based on $R_{T'_n}$ that (exactly) leads from W_n to Z_n.*

Proof. The proof mimics exactly the proof of Lemma 6.1. ■

Theorems 6.1 and 6.2 say that as long as either the number of predicate symbols or the number of constants in the signature is not bounded in advance, then goal reachability is PSPACE-hard. As the next corollary shows, this implies the PSPACE-hardness of both versions of the collaborative planning problem with compliance to a confidentiality policy.

Corollary 6.1 *Both the collaborative planning problem with system compliance and the collaborative planning problem with plan compliance are* PSPACE*-hard.*

Proof. The embeddings given in the proofs of Theorems 6.1 and 6.2 are into systems with a single agent whose confidentiality policy does not view any configuration as being critical. Thus the hardness is already achieved in the single-agent case where both system compliance and plan compliance are vacuously true. ■

6.2 Upper Bounds

In determining the upper complexity bounds, we will be very explicit about what parameters contribute to the complexity, as well as how they are represented. In both versions of the problem the complexity should be with respect to the size of the input. The problem has four inputs: 1) the transition system itself, given by the (well-balanced) actions available to the agents, 2) the initial configuration of the system, 3) the (partial) goal configurations, and 4) the data confidentiality policies of each agent.

We begin with the initial configuration of the system. The size of its description depends on two main factors. The first is the number of facts (including multiplicity) that may be present in a configuration. Since we are using well-balanced actions this will be a fixed number m. The second factor is the size of a description of a fact. If we let S_T denote the number of facts expressible in the finite signature Σ, then this will be on the order of $\log_2(S_T)$. Thus a direct binary encoding of a configuration can be done in $O(m \cdot \log_2(S_T))$ space, since each fact requires $O(\log_2(S_T))$ space and each configuration has m facts. Similarly, each action can be encoded in $O(2m \cdot \log_2(S_T))$ space since an applicable action is just a list of two configurations each requiring space $O(m \cdot \log_2(S_T))$.

Now let us discuss the representation of the transition system itself. A simple way of describing the system is by simply listing each propositional instance of an action. However, any complexity result which is derived with respect to the size of this list can be misleading. The problem is that while each instance may have a compact representation, the list may be very long. In fact, the length of the list is likely to be exponential with respect to the length of a list of first order actions (actions in which free variables are implicitly universally quantified). In this case, a PSPACE result with respect to the size of the propositional list of actions hides the fact that the list is already exponential with respect to another reasonable description of the system. The PSPACE result is therefore somehow dishonest. The polynomial is already with respect to an exponential quantity, and so the real result is EXPSPACE.

Instead of using this straightforward, but inefficient, representation of the actions, we will use the smallest representation that is possible. Our idea is to use the Kolmogorov descriptive complexity of the set of actions. This amounts to representing the actions by a program which recognizes valid instances. The complexity result we obtain will be independent of the specific program we use. In particular, it will work for the small program that results from the definition of Kolmogorov complexity [**?**]. In this way, the PSPACE result implies that the space needed to perform the computation is polynomial with respect to an optimally small representation of the system. The result is therefore honest, in that it does not hide any exponential expansion which arises from inefficient representations of the inputs.

We will use this Kolmogorov descriptive complexity for the other three inputs. The actions of the transition system will be represented by a program $\mathcal{T}$ which recognizes valid instances of actions. That is, $\mathcal{T}(r) = 1$ if $r \in R_T$ and $\mathcal{T}(r) = 0$ otherwise. Similarly, the (partial) goal configurations will be represented by a program $\mathcal{G}$ which recognizes the global configurations which contain one of the partial goals. Finally, the data confidentiality policies will also be represented by a program $\mathcal{C}$ which recognizes global configurations which are critical according to at least one agent's policy. In order for the PSPACE result to hold, we must assume that each of these programs runs in PSPACE with respect to the size of their inputs. For Theorems 6.5 and 6.6 we will need to assume that these programs run in polynomial time.

Although both versions of the collaborative planning problem with compliance are stated as decision problems, we prove more than just PSPACE decidability. Ideally we would also be able to generate a plan in PSPACE when there is a solution. Unfortunately, the number of actions in the plan may already be exponential in the size of the inputs, precluding PSPACE membership of plan generation. For this reason we use the notion of "scheduling" a plan in which an algorithm will also take an input i and output the i-th step of the plan.

Definition 6.1 *An algorithm is said to* schedule *a plan if it*

(i) *finds a plan if one exists, and*

(ii) *on input i, if the plan contains at least i actions, then it outputs the i-th action of the plan otherwise it outputs $\perp$.*

Before presenting the main theorems we include a lemma which gives bounds on S_T the number of facts expressible in a signature, and also on $L_T(m)$ which denotes the number of configurations with m or fewer facts expressible in a signature. We omit the proof as it is purely combinatorial and does not provide great insight to the concepts presented in this paper.

Lemma 6.3 *Given a local state transition system T over a finite signature Σ,*

(i) *let S_T be the total number of facts in signature Σ, and*

(ii) *let $L_T(m)$ denote the total number of configurations in signature Σ whose number of facts (counting repetitions) is bounded by m.*

Then

(a) $S_T \leq (J \cdot D^a)$, *where:*

(a1) J *is the total number of predicate symbols in signature* Σ,

(a2) a *is an upper bound of their arity, and*

(a3) D *is the total number of constants in signature* Σ.

(b) *For any* m*:*

$$L_T(m) \leq \binom{m + S_T}{m} = \frac{(m + S_T)!}{m! \cdot (S_T)!} \leq \min\{(S_T + 1)^m, (m + 1)^{S_T}\}.$$

We are now ready to demonstrate the complexity of the two versions of the planning problem with confidentiality. The following theorem says that the problem of determining system compliance and scheduling a plan is in PSPACE.

Theorem 6.3 *There is an algorithm applicable to the following inputs:*

(i) *a program* $\mathcal{T}$ *which recognizes (in* `pspace`*) the actions of a local state transition system* T *with actions* R_T,

(ii) *an initial configuration* W,

(iii) *a program* $\mathcal{G}$ *which recognizes (in* `pspace`*) global configurations which contain at least one of the partial goals* $Z_1, \ldots, Z_k$,

(iv) *a program* $\mathcal{C}$ *which recognizes (in* `pspace`*) global configurations which contain at least one of the critical configurations which violate a policy, and*

(v) *a natural number* $0 < i \leq L_T(m)$

which behaves as follows:

(a) *If all critical configurations are unreachable from* W*, and if there is a plan leading from* W *to some* Z_i*, then it outputs "yes" and schedules the plan, otherwise it outputs "no".*

(b) *It runs in* `polynomial space` *with respect to* $|\mathcal{T}|$, $|W|$, $|\mathcal{G}|$ *and* $|\mathcal{C}|$.

Proof. The proof will rely on several critical facts about PSPACE. Namely, we rely on the equivalence of PSPACE, NPSPACE and COPSPACE [**?**].

Policy Compliance:

The algorithm first checks that none of the critical configurations are reachable from W. That is, we must check that every reachable configuration Z satisfies $\mathcal{C}(Z) = 0$. To do this, we give a nondeterministic algorithm which accepts exactly when a critical configuration *is* reachable. We then apply Savitch's Theorem [**?**] to determinize the algorithm. Finally, we swap the accept

and reject conditions to get a deterministic algorithm which accepts exactly when *all* critical configurations are *unreachable*.

We begin with $W_0 := W$. For any $t \geq 0$, we first check if $\mathcal{C}(W_t) = 1$. If so, then we return "yes". Otherwise we proceed to guess an action $r : XX' \rightarrow_A YY'$ such that $\mathcal{T}(r) = 1$. If no such action exists we return "no". If such an action does exist then we let W_{t+1} be the configuration which results from applying action r to configuration W_t, and we replace W_t by W_{t+1}. We can show that a critical configuration is reachable if and only if there is a sequence of at most $L_T(m)$ guesses that ends in a critical configuration. This means we should stop guessing actions and return "no" after having already guessed $L_T(m)$ of them.

It is clear that if a critical configuration is reachable from W then there is a sequence of guesses which will result in the algorithm returning "yes". Otherwise the algorithm will always return "no". Thus the algorithm correctly decides the reachability of critical configurations. It only remains to show that this algorithm uses space which is polynomial in $|\mathcal{T}|$, $|W|$, and $|\mathcal{C}|$.

Let us now determine the space required for each t-th step. We need to record the configuration W_t. Since the configuration size never grows, it follows that $|W_t| = O(|W|)$ for any configuration W_t. Program $\mathcal{C}$ can verify $\mathcal{C}(W_t) = 1$ in space polynomial in $|W_t|$ by the assumption about the program $\mathcal{C}$. Our algorithm will need to keep track of $\mathcal{C}$'s functioning which can be done in space which is polynomial in $|\mathcal{C}|$. We then need to record the action r which is guessed. In the worst case an action r will have m facts in both the pre- and post-conditions. This amounts to representing two complete configurations which can be done in $O(|W|)$ space. Program $\mathcal{T}$ can verify $\mathcal{T}(r) = 1$ in space which is polynomial in $|r|$. Since $|r| = O(|W|)$ this verification can be done in space polynomial in $|W|$. Our algorithm will need to keep track of $\mathcal{T}$'s functioning which can be done in space which is polynomial in $|\mathcal{T}|$. In replacing W_t by W_{t+1} we may have to record two configurations at once, but we never record more than two. Finally, we should keep track of how many actions we have guessed so far. Since the algorithm stops after guessing $L_T(m)$ actions, this can be maintained on $\log_2(L_T(m))$ space. By Lemma 6.3, this is less than $m \cdot \log_2(S_T + 1)$ which we already saw is $O(|W|)$. Since the space at each step is polynomial in $|\mathcal{T}|$, $|W|$ and $|\mathcal{C}|$, the total space is also polynomial in those parameters. Thus the problem is found to be in NPSPACE. As stated above, we can determinize this algorithm and swap the accept and reject conditions to obtain the algorithm we are looking for, which remains in PSPACE.

Scheduling the Plan:

We now have to give an algorithm to schedule a plan if it exists, and show this algorithm is also in PSPACE. This is easily done by adapting the algorithm given above. We can replace every occurrence of $\mathcal{C}$ with $\mathcal{G}$, and we get a (nondeterministic) algorithm which decides the existence of a plan leading from W to one of the goals $Z_1, \ldots, Z_k$. In order for the algorithm to schedule a plan (and not just decide the existence of a plan) the algorithm will remember the guess for the i-th action. This only adds $O(|W|)$ space to the complexity. Thus this algorithm will run in space which is polynomial in $|\mathcal{T}|$, $|W|$ and $|\mathcal{G}|$. This time we only need to determinize the algorithm to conclude that it is in PSPACE.

Combining the Algorithms:

Finally, we note that we can combine these two algorithms in sequence to get the correct behavior. If either the first or the second parts fail, then we output "no". Otherwise we output "yes" along with the i-th action of the plan. This combined algorithm will run in space which is polynomial in $|\mathcal{T}|$, $|W|$, $|\mathcal{G}|$ and $|\mathcal{C}|$. ■

The previous theorem combines with Corollary 6.1 to imply that the collaborative planning problem with system compliance is PSPACE-complete. Due to the use of data confidentiality policies, this is actually a generalization of our PSPACE result in [**?**]. The following corollary is the formal statement of this fact. (We modify the bound on k, the number of partial goals in order to keep the input itself within the size bound.)

Corollary 6.2 *There is an algorithm applicable to the following inputs:*

(i) *a program $\mathcal{T}$ which recognizes (in* `pspace`*) the actions of a local state transition system T with actions R_T,*

(ii) *an initial configuration W,*

(iii) *a list $Z_1, \ldots, Z_k$ of partial goal configurations for $k = O(poly(|W|)$.*

which behaves as follows:

(a) *If the system preserves the privacy of all agents according to Definition 4.3, and if there is a plan leading from W to some Z_i, outputs "yes", otherwise it outputs "no".*

(b) *It runs in* `polynomial space` *with respect to $|\mathcal{T}|$, and $|W|$ (which is $O(m \cdot \log_2(S_T))$).*

Proof. The algorithm is essentially the same as in the previous proof. The key difference is that there is no policy which is input to the problem. Instead, the policy is implicitly encoded in the initial configuration. We can replace the program $\mathcal{C}$ with a process that looks at each term in the current configuration and determines if it is one of the terms that should be protected according to Definition 4.3. It then determines if the current occurrence of the term violates Definition 4.3 by looking at who owns the predicate in which it appears. This is easily done in space which is polynomial in the size of a configuration (*i.e.* in $|W|$).

Notice also that the complexity is no longer with respect to $|\mathcal{G}|$, since we input an explicit encoding of the partial goals $Z_1, \ldots, Z_k$. Essentially, we replace $\mathcal{G}$ with this list. At each step we can check the current configuration against those partial goals on the list in space which is polynomial in $|W|$. Assuming $k = O(\text{poly}(|W|))$ is a special case of assuming $|\mathcal{G}| = O(\text{poly}(|W|))$, which allows us to put the polynomial complexity only in terms of $|W|$. ■

Intuitively, the corollary follows because the policy can be extracted from the initial configuration in polynomial space, and because we restrict the explicit representation of the partial goals

as a list to be small. By using data confidentiality policies we gain a great deal of expressiveness at the small cost of an extra input parameter. Similarly, by considering more possibilities for the partial goal configurations we must pay a small cost. The corollary is just a special case in which $|\mathcal{G}|$ and $|\mathcal{C}|$ are essentially restricted to be polynomial in $|W|$. Thus these extra parameters only come into play when their size greatly exceeds that of $|W|$.

The next theorem states that the problem of scheduling a compliant plan is also in PSPACE.

Theorem 6.4 *There is an algorithm applicable to the following inputs:*

(i) *a program $\mathcal{T}$ which recognizes (in* `pspace`*) the actions of a local state transition system T with actions R_T,*

(ii) *an initial configuration W,*

(iii) *a program $\mathcal{G}$ which recognizes (in* `pspace`*) global configurations which contain at least one of the partial goals $Z_1, \ldots, Z_k$,*

(iv) *a program $\mathcal{C}$ which recognizes (in* `pspace`*) global configurations which contain at least one of the critical configurations which violate a policy, and*

(v) *a natural number $0 < i \leq L_T(m)$*

which behaves as follows:

(a) *If there is a plan leading from W to some Z_i which avoids critical configurations, then it outputs "yes" and schedules the plan, otherwise it outputs "no".*

(b) *It runs in* `polynomial space` *with respect to $|\mathcal{T}|$, $|W|$, $|\mathcal{G}|$ and $|\mathcal{C}|$.*

Proof. We can adapt the proof of Theorem 6.3 for our situation here. At each step, the nondeterministic algorithm first checks if $\mathcal{C}(W_t) = 1$. If so, then the algorithm outputs "no". Otherwise it checks if $\mathcal{G}(W_t) = 1$. If so, then it outputs "yes". If not, then it guesses an action r in order to generate the next configuration W_{t+1}. In this way, we only continue along a path if it avoids the critical configurations. As we did above, we can remember the i-th action using space which is $O(|W|)$. Since we can store each configuration and action in PSPACE as well as perform the necessary checks in PSPACE, we again find that the entire procedure runs in PSPACE with respect to the size of the inputs. Finally, we determinize the algorithm according to the proof of Savitch's Theorem [**?**]. ■

This theorem combines with Corollary 6.1 to imply that the collaborative planning problem with plan compliance is PSPACE-complete.

We end this section by considering the complexity of the planning problems with confidentiality under one further restriction. Instead of viewing the signature Σ of the transition system as an input to the problem, we fix it *in advance*. This may be a reasonable thing to consider if the agents involved participate in distinct yet related collaborations on a periodic basis. For example, a group of companies may wish to perform quarterly collaborative forecasting. The language of

the problem can be viewed as a fixed aspect of the system instead of an input parameter. Under this assumption we show that both versions of the collaborative planning problem are solvable in polynomial time with respect to the size of their input.

Although the time complexity of the problems is exponential, the following theorems also serve to isolate the source of the exponential complexity namely, the number of facts expressible by the signature. In practice, this means that the polynomial time bound will actually be some polynomial of a large (but fixed) degree.

Theorem 6.5 *Let Σ be a fixed finite signature (consisting of a finite number of predicate symbols with their arity and of a finite number of constants).*
Then there is an algorithm β_Σ applicable to the following inputs:

(i) *a program $\mathcal{T}$ which recognizes, in polynomial time, the actions of a local state transition system T with actions R_T,*

(ii) *an initial configuration W,*

(iii) *a program $\mathcal{G}$ which recognizes, in polynomial time, global configurations which contain at least one of the partial goals $Z_1, \ldots, Z_k$,*

(iv) *a program $\mathcal{C}$ which recognizes, in polynomial time, global configurations which contain at least one of the critical configurations which violate a policy*

which behaves as follows:

(a) *If all critical configurations are unreachable from W, and if there is a plan leading from W to some Z_i, then it outputs the plan, otherwise it outputs "no".*

(b) *It runs in* `polynomial time` *with respect to $|\mathcal{T}|$, $|W|$, $|\mathcal{G}|$, and $|\mathcal{C}|$.*

Proof. The crucial fact to keep in mind throughout this proof is that the total number of configurations is $L_T(m) \leq (m+1)^{S_T}$. Since we have fixed a signature *in advance*, S_T is viewed as a constant. Thus $L_T(m)$ is polynomial in m. Similarly, $|W| = O(m \cdot \log_2(S_T)) = O(m)$, so we find that $L_T(m)$ is polynomial in $|W|$.

The algorithm works the same as the determinized algorithm in the proof of Theorem 6.3. The size of the reachability tree is polynomial in $L_T(m)$. Thus the number of steps is polynomial in $|W|$. Since we assume that each of $\mathcal{T}$, $\mathcal{G}$ and $\mathcal{C}$ run in polynomial time (with respect to $|W|$), the whole algorithm now runs in polynomial time with respect to the size of the four input parameters. ■

Theorem 6.6 *Let Σ be a fixed finite signature (consisting of a finite number of predicate symbols with their arity and of a finite number of constants).*
Then there is an algorithm β_Σ applicable to the following inputs:

(i) *a program $\mathcal{T}$ which recognizes, in polynomial time, the actions of a local state transition system T with actions R_T,*

(ii) *an initial configuration W,*

(iii) *a program $\mathcal{G}$ which recognizes, in polynomial time, global configurations which contain at least one of the partial goals $Z_1, \ldots, Z_k$,*

(iv) *a program $\mathcal{C}$ which recognizes, in polynomial time, global configurations which contain at least one of the critical configurations which violate a policy*

which behaves as follows:

(a) *If there is a compliant plan leading from W to one of the goal configurations $Z_1, \ldots, Z_k$, then it outputs the plan, otherwise it outputs "no".*

(b) *It runs in* `polynomial time` *with respect to $|\mathcal{T}|$, $|W|$, $|\mathcal{G}|$, and $|\mathcal{C}|$.*

Proof: This algorithm works the same as the determinized algorithm in the proof of Theorem 6.4. Since the reachability tree is polynomial in $L_T(m)$ which is polynomial in $|W|$, we can conclude that the whole algorithm runs in polynomial time with respect to the size of the four input parameters. ■

7 Related Work

Our formalism shares a lot of similarities with the Multiset Rewriting formalism (MSR) presented in [**?**, **?**], which is a notable example of the use of state transition systems. One key difference with our work is in the treatment of the adversary. In the context of collaboration each participant views all the others as potential colluding adversaries. In particular, the representation of security protocols in the MSR formalism [**?**] assumes that honest participants act deterministically and are computationally very limited, while the adversary has unbounded memory and can act nondeterministically. Here, each participant can make use of nondeterminism, and we place no fixed bound on the memory of the participants.

Although there is no fixed bound on the agents' memory, using well-balanced actions imposes an implicit bound on their memory. It is possible that this notion of bound could be modeled directly in MSR by encoding as many sessions of the protocol as necessary for a single agent, thus given the agent as much memory as desired up to an arbitrary bound.

Complexity results are known for many types of Petri nets with various restrictions [**?**]. In [**?**, **?**] we demonstrated formal translations between the coverability problem for Petri nets and the reachability problem for local state transition systems. Under our translation, well-balanced local state transition systems do not seem to correspond exactly to any of the various types of Petri nets described in [**?**].

Our focus on data flow which violates a confidentiality policy is somewhat reminiscent of other information flow formalisms such as [**?**, **?**, **?**, **?**]. These formalisms tend to focus on the trace of an execution which is observable to a (usually passive) adversary. Confidentiality leaks occur in these formalisms when the adversary can successfully determine (through implicit inference) the initial assignment to some variable. This differs from our state-based approach in which the agents' actions are explicitly modeled.

There has been other work on privacy in settings where data sharing is inevitable. One formalism, Contextual Integrity (CI) [**?**, **?**], is a philosophical framework which views privacy as a right to the appropriate flow of information. In order to determine what is appropriate, several parameters such as contexts, roles, attributes, and transmission principles are considered. CI has a formal instantiation which has its foundation in temporal logic allowing it to express notions of time which are crucial for the instantiation of the philosophical framework. It may be possible to map our formalism onto a fragment of the CI formalism in certain types of scenarios.

On the more applied side of the scale, there are numerous works about how to achieve confidentiality in specific settings such as scientific data sharing [**?**, **?**], and the controlled release of medical information [**?**]. Additionally, there is work analyzing the privacy loss of specific distributed constraint optimization algorithms [**?**, **?**]. These approaches tend to address particular details of the situation in question, and therefore lack the abstractness necessary for capturing a wide range of scenarios. However, both of [**?**, **?**] apply general metrics for privacy to collaborative scheduling algorithms. Our formalism currently lacks a way of quantifying the interplay between confidentiality and goal reachability.

8 Conclusion and Future Work

In this paper we have presented an abstract model for collaboration which addresses the inherently competing notions of protecting and releasing resources. We have discussed what it means to generate a collaborative plan and maintain the participants' confidentiality of information or resources relative to a data confidentiality policy. We have demonstrated the logical foundation of our approach with a translation into affine logic which relates the existence of a collaborative plan to derivability. We have shown that deciding the existence of a well-balanced collaborative plan with both system compliance and plan compliance is PSPACE-complete. We saw that by fixing in advance the number of constants and predicates, these problems are solvable in polynomial time.

Currently we have considered systems with a finite signature that remains unchanged throughout an execution. In practice, agents are continually creating and destroying values. We would like to consider ways to model the generation of fresh values. This would be appropriate when, for example, creating new passwords. The ability to create new values leads naturally to a more thorough investigation of our formalism's ability to distinguish between current and obsolete knowledge. Can our notion of confidentiality policy easily accommodate this distinction?

While data confidentiality policies specify where certain data may or may not go, it does not

explicitly specify rules of transmitting the data. For example, the results of some medical tests may be known by both the doctor and the nurse, but only the doctor is allowed to pass the results on to the patient. We hope to integrate this type of information flow policy into our work. Such policies may also provide us with the ability to trace information leaks back to their source. This would provide a sort of auditing mechanism.

The interplay between confidentiality and goal reachability suggests agents who make rational trade-offs between the two. We may be able to gain more insight into the situation by enriching our formalism with an explicit notion of rationality. We would like to consider such "rational adversaries" both in contrast to and in combination with malicious adversaries.

It would be interesting to see how our worst-case complexity results actually manifest on real examples. It would be useful to have an implementation of the algorithms that would allow us to explore more realistic complexity results.

This work and all the suggestions for future work can also be put into a synchronous context in which there is an explicit notion of time. The existence of such a global clock may affect the properties of systems in interesting ways.

Acknowledgements

Thanks to Anupam Datta, Zack Ives, Pat Lincoln, John Mitchell, Helen Nissenbaum and Steve Zdancewic and other members of the Penn Computer Security Seminar for their useful suggestions and comments. We also thank the anonymous referees for useful and insightful comments that helped us clarify a number of ideas.

A Some Rules of Affine Logic

Structural rules

$$\frac{}{\Gamma; A_1, \ldots, A_n \vdash A_i}\ \text{id} \qquad \frac{\Gamma, A; \Delta, A \vdash C}{\Gamma, A; \Delta \vdash C}\ \text{clone}$$

Cut rules

$$\frac{\Gamma; \Delta_1 \vdash A \qquad \Gamma; \Delta_2, A \vdash C}{\Gamma; \Delta_1, \Delta_2 \vdash C}\ \text{cut} \qquad \frac{\Gamma; \cdot \vdash A \qquad \Gamma, A; \Delta \vdash C}{\Gamma; \Delta \vdash C}\ \text{cut!}$$

Left rules	**Right rules**
$\dfrac{\Gamma; \Delta \vdash C}{\Gamma; \Delta, \mathbf{1} \vdash C}\ \mathbf{1}\ell$	$\dfrac{}{\Gamma; \cdot \vdash \mathbf{1}}\ \mathbf{1}r$
$\dfrac{\Gamma; \Delta, A_1, A_2 \vdash C}{\Gamma; \Delta, A_1 \otimes A_2 \vdash C}\ \otimes\ell$	$\dfrac{\Gamma; \Delta_1 \vdash C_1 \qquad \Gamma; \Delta_2 \vdash C_2}{\Gamma; \Delta_1, \Delta_2 \vdash C_1 \otimes C_2}\ \otimes r$
$\dfrac{\Gamma; \Delta_1 \vdash A \qquad \Gamma; \Delta_2, B \vdash C}{\Gamma; \Delta_1, \Delta_2, A \multimap B \vdash C}\ \multimap \ell$	$\dfrac{\Gamma; \Delta, A \vdash C}{\Gamma; \Delta \vdash A \multimap C}\ \multimap r$
$\dfrac{\Gamma; \Delta, A \vdash C}{\Gamma, \Delta, \exists x.A \vdash C}\ \exists\ell$	$\dfrac{\Gamma; \Delta \vdash [t/x]C}{\Gamma; \Delta \vdash \exists x.C}\ \exists r$