Collaborative Planning With Privacy[*]

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

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

Abstract

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 private/secret are two competing notions which affect the outcome of a collaboration. This paper proposes a formal model of collaboration which addresses privacy/secrecy concerns. We draw on the notion of a plan which originates in the AI literature. We consider transition systems in which actions have pre- and post-conditions of the same size. We show it is PSPACE*-complete to decide whether a given such system protects the privacy/secrecy of its participants and whether it contains a plan leading from a given initial state to a desired goal state.*

1. Introduction

With information and resources becoming more distributed interaction with external services is becoming more necessary. There is a lot of software available which is designed to foster collaboration through communication, resource sharing and data sharing. Collaborating agents are usually viewed to have a friendly 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 privacy/secrecy 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.

Privacy/secrecy in collaboration doesn't only apply to personal information. It is common for documents to be written by two or more authors from distinct organizations. For instance a position paper regarding computer industry practices may be written by individuals representing different companies or government agencies. These individuals and the organizations they represent have a clear, common goal of publishing their joint position. Each author might know some relevant information which is a company secret. They are therefore unwilling to share it with the others. What each author contributes to the paper may be viewed as a sanitized version of his or her secret knowledge. This knowledge is transformed in some way that allows it to be published. For example, an author might provide general information about how a proprietary algorithm works while removing details that would allow others to duplicate it. This type of situation often arises in the discussion of multilevel security (MLS or MILS) and non-interference [23, 11, 27].

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

[*]Research partially supported by OSD/ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing" through ONR Grant N00014-01-1-0795 and OSD/ONR CIP/SW URI "Trustworthy Infrastructure, Mechanisms, and Experimentation for Diffuse Computing" through ONR Grant N00014-04-1-0725. Additional support from NSF Grants CNS-0429689 and CNS-0524059.

dynamic and growing biological data (see [29, 13]). 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 private/secret data. Research groups may have informal policies, or practices that determine when they are willing to incorporate outside data into their results.

So far we have discussed privacy/secrecy only with respect to information. However, it can also apply to sensitive material resources. Consider a chemical plant which has various sectors designed to keep chemicals physically separate for safety reasons. While separation is clearly necessary, the chemicals must also be distributed from time to time in prescribed amounts. There should be strict guidelines restricting the conditions under which chemicals are distributed, and their quantity in transit.

Privacy/secrecy concerns are present in all of these situations. 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 main privacy/secrecy concern is that the private (High) data of a participant might be learned by the others from the data which is released (Low data).

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 private 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 private or High facts) together with some shared group state (comprised of group/public or Low facts). In fact, we use several distinct High labels, with one for each agent, and a single Low label. 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.

We consider systems with well-balanced transitions which have the same number of facts in both the preconditions and the post-conditions. Intuitively, this means that each agent has a private (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.

We find the complexity of determining the existence of a collaborative plan with privacy/secrecy. Our focus is on the interplay between two concerns: Can we demonstrate the existence of a collaborative plan, and can we do so while providing some privacy guarantees to the participants? Since the information which is released is considered safe to distribute, the main privacy concern is that the other collaborators could collectively learn the secret information of some participant. If a piece of information is secret at the beginning of a collaboration then a local state transition system protects that information if it is never released throughout the course of any plan.

Our main complexity results (Corollary 6.3 and Corollary 6.6) combine to say that given a local state transition system, and given an initial configuration W and a goal configuration Z, the problem of deciding

(a) if a collaborative plan exists which leads from W to the goal Z, and
(b) if the system can never release the private (High) information of any participant

is PSPACE-complete. We also show in Corollary 6.5 that if we fix in advance the size of the domain and the number of relations considered, that the above problem can be solved in polynomial time.

We also demonstrate the logical foundation of our approach in linear logic [9, 14]. 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 (see *e.g.*, [14, 20, 1, 26]). On the other hand, multiset rewriting formalisms have proved useful in the analysis of security protocols (*e.g.*, [5, 2]) as well as in the field of computation (*e.g.*, [19, 16, 17, 15, 18]).

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 (*e.g.*, [14, 20, 26]).

In this example there are three blocks represented by the costants 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. That is, $\alpha(Z \cup pre(\alpha)) = Z \cup post(\alpha)$, where $Z \cap pre(\alpha) = \emptyset$, and $Z \cap post(\alpha) \subseteq pre(\alpha)$. 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

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

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 closed, 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 [14, 20], 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. As a starting point, we restrict our terms to be constants and variables. In the future, we would like to consider a richer language of terms.

A signature Σ consists of predicate symbols with their arity, and many sorted constants. As above, a fact is a closed, 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 private/secret and public/group has the structure of a simple tree. There is a separate High label for each of the participants and there is a single Low label as illustrated by the following figure:

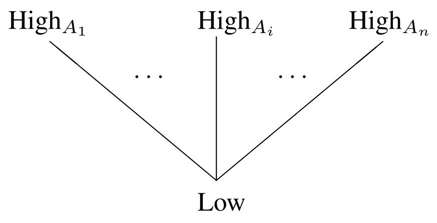

Figure 1. Security Levels.

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 $XX' \rightarrow YY'$. 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 $X' \rightarrow_A Y'$ 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 $r : XX' \rightarrow_A YY'$ 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

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

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 $r : XX' \rightarrow_A YY'$, 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 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

Consider a collaboration with two agents A and B. They each have only one transition rule. The rules are

$$r_1 : X_A X' \rightarrow_A Y_A Y' \text{ and}$$

$$r_2 : W_B Y' \rightarrow_B U_B Z'$$

which belong to agents A and B respectively.

Now $R = \{r_1, r_2\}$. If the initial state is $S = X_A X' W_B$ then we can reach the state $V = Y_A U_B Z'$ as follows.

$$X_A X' W_B \rhd_{r_1} Y_A Y' W_B \rhd_{r_2} Y_A U_B Z'$$

We see that $S \rhd_R^* V$, and also that $S \rightsquigarrow_R^* Z'$. In this example it is evident that collaboration is necessary in order to reach the partial goal Z' from the initial system state S.

4. Privacy

When entering into a collaboration the participants must balance the importance of reaching a common goal against the importance of protecting their own information. If Alice's information is too sensitive, then she will be unwilling to collaborate if she knows that every plan which reaches the goal must leak that information. On the other hand, maybe Alice considers the goal to be more important. She still wants to keep her information secret if possible. However, if she knows the goal can only be reached by releasing some sanitized version of her data, then she might be willing to make the trade-off.

In order to express these notions in a formal way, we must define what it means for secret information to leak. While there may be a number of ways of representing an information leak, as a starting point here we choose the most explicit form. Namely, if in the initial configuration of the system, a term t occurs only in Alice's private (High) predicate (as $P_A(t, \bar{s})$ for any tuple of terms $\bar{s}$), then we say that t is *learned* by Bob if, in a later configuration, t occurs in some public (Low) predicate, or a private (High) predicate of Bob (*i.e.*, either $Q'(t, \bar{u})$ or $Q_B(t, \bar{u})$ for some tuple of terms $\bar{u}$).

The action $P_A(t) \rightarrow P'(t)$, where P' is a public predicate, provides a simple way for releasing t. However, Bob may be able to learn Alice's private (High) information by more indirect means by combining public (Low) information she releases and his own private (High) data.

Example 4.1

Let Alice's actions include

$$r_1 : P_A(15\_A\_Pwd)S_A(7\_B\_Share) \\ \rightarrow P_A(15\_A\_Pwd)S_A(7\_B\_Share)P'(8\_A\_Share)$$

and let Bob's actions include

$$r_2 : Q_B(7\_B\_Share)P'(8\_A\_Share) \rightarrow Q_B(15\_A\_Pwd)$$

When $R = \{r_1, r_2\}$, we find that

$$P_A(15\_A\_Pwd)S_A(7\_B\_Share)Q_B(7\_B\_Share) \\ \rightsquigarrow^*_R Q_B(15\_A\_Pwd)$$

which means that Bob is able to learn Alice's password, 15_A_Pwd. Notice that, in this example, the password 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 password.

While this example was written as an illustration of secret sharing, similar situations may occur without Alice intending it to. In that case Alice's private (High) information is not protected from Bob. The following definition says what it means for a system to protect an agent's private (High) information.

Definition 4.1 *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.*

That is to say, A's privacy is protected if for every term t, whenever t occurs only in A's private (High) predicates in the initial state, then no plan can ever lead to a fact of the form $Q'(t, \bar{u})$ or $Q_B(t, \bar{u})$ for some $\bar{u}$. This is a global condition on the space of all possible plans starting from a given initial state.

While the classical planning problem asks only for a plan which leads from the initial state to the goal state, our situation demands more. We would like to find a plan which leads from the initial state to the goal state, but we also make sure that the transitition system protects the privacy of all participants. We would like to ask the following question:

The Collaborative Planning Problem with Privacy:
Given a local state transition system and given an initial state W and a partial goal Z, does there exist a plan which leads from W to Z, and does the system protect the privacy of all agents?

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 has the same complexity as the reachability problem for Petri nets [8, 19]. Although this is decidable [21], 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 [14]. A *well-balanced* transition is a transition which has the same number of facts in the pre-condition and the post-condition (including repetitions). In well-balanced local state transition systems, the total number of facts present in the global state remains constant.

While at first this may seem like a big restriction, in practice we are still able to model most scenarios 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 limit on the number of fields we may choose. Informally, in well-balanced systems, agents only update their databases, while in general systems they may create new fields.

To demonstrate this, we transform the example from earlier in this section, 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 the privacy condition (Definition 4.1) does not refer to $*$. Alice's action may then be rewritten in the following well-balanced form.

$$\begin{aligned} r_1 : & P_A(15\_A\_Pwd)S_A(7\_B\_Share)P'(*) \\ & \to P_A(15\_A\_Pwd)S_A(7\_B\_Share)P'(8\_A\_Share) \end{aligned}$$

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

$$\begin{aligned} r_2 : & Q_B(7\_B\_Share)P'(8\_A\_Share) \\ & \to Q_B(15\_A\_Pwd)P'(*) \end{aligned}$$

In this way, if the initial configuration has n 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 n. 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 n.

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.

4.1 Remarks

In modeling a real world collaboration the agents may have a given protocol in mind which they would like to follow. Each agent would then have actions which model their part of the protocol. The actions and initial configuration then determine a space of collaborative plans, or protocols, which contains the protocol the agents had in mind. A solution to the collaborative planning problem with privacy does two things. First it determines that the entire space of protocols satisfies Definition 4.1. Secondly, it picks a plan which achieves the goal.

By determining that the system protects the privacy/secrecy of all agents, each agent has a guarantee that even if the other participants don't follow the plan, or if they perform extra private (High) computations, the secrets will not leak. This is because any deviation from the agreed upon plan will be another plan which also satisfies Definition 4.1.

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* private information? To illustrate this point let us return, once again, to the Example 4.1 above. Alice's action may not only release enough information to reveal her password, it may actually change her password in the process. The action may be changed to the following:

$$\begin{aligned} r_1 : & P_A(15\_A\_Pwd)S_A(7\_B\_Share)P'(*) \\ & \to P_A(21\_A\_Pwd)S_A(7\_B\_Share)P'(8\_A\_Share) \end{aligned}$$

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

Bob would still be able to learn Alice's *old* password, but he has no access to her *current* password. Her old password 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 password. This may provide him access to more sensitive data. Currently, Definition 4.1 does not address this issue. For the moment we simply remark that such notions are expressible in our formalism, and we leave a more complete investigation of them for future work.

5. Foundation in Logic

In this section we present the logical foundation of our approach. It is 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. Here we demonstrate a connection to a variation of linear logic known as affine logic.

Linear logic (LL), which was introduced in [9], 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, the 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.

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

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 where 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 [3]). 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.*

Proof Outline. The proof is by induction on the length of the transition chain. The base case is trivial. To prove the induction step, it suffices to prove that if $W \rightsquigarrow_r Z$ then $\ulcorner r \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash \ulcorner I \urcorner \otimes \ulcorner Z \urcorner$ is derivable in affine logic. This is done by bringing $\ulcorner r \urcorner$ into the linear context using the clone rule, and then isolating the antecedent using the $\multimap$-ℓ rule. The remaining rules help in reducing the result to logical axioms. ∎

Theorem 5.2 (Completeness) *For every pair of states W, Z and every rule set R_T defining a paricipant 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 $W \rightsquigarrow^*_{R_T} Z$.*

Proof Outline. This proof relies on the fact that we are able to transform an arbitrary deduction in affine logic of $\ulcorner R_T \urcorner; \ulcorner I \urcorner \otimes \ulcorner W \urcorner \vdash \ulcorner I \urcorner \otimes \ulcorner Z \urcorner$ into a normal form similar to the form which results in the proof of soundness [2, 3, 17]. When this is done, we can then take the actions which correspond to the linear implications used in the $\multimap$-ℓ rules. By listing them from the root of the derivation upwards, we create the desired transition sequence. ■

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, $\mathtt{Ax}_\mathtt{T}$, 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 $\ulcorner I \urcorner \otimes W \vdash Z \otimes V$ is derivable in the LL-theory for some V (see [14]).

The privacy condition discussed in Section 4 requires that for every term t which, in the initial configuration W, occurs only in private predicates of A, no plan can lead from W to a fact of the form $Q_B(t, \bar{u})$ or $Q'(t, \bar{u})$. This is equivalent to the requirement that

$$\ulcorner I \urcorner \otimes W \vdash \exists y_1, \ldots, \exists y_n Q_B(t, y_1, \ldots, y_n) \text{ and}$$

$$\ulcorner I \urcorner \otimes W \vdash \exists z_1, \ldots, \exists z_n Q'(t, z_1, \ldots, z_n)$$

are not provable in the AL-theory corresponding to the given system. This is because of the so called *existence property* of linear and affine logics which says that the sequents of this form are provable only by instantiating the existential quantifiers [9].

6. Complexity

6.1. Lower PSPACE Bounds

We start with the simulation of 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 $x_1x_2..x_n$, is written in cells 1, 2,..,n on the tape. In addition, a special marker # is written in the $(n+1)$-th cell.

\$	x_1	x_2	$\cdot$	$\cdot$	$\cdot$	x_n	#				$\cdots$

(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 $i = 0, 1, .., n+1$, 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 $j = 0, 1, .., n+1$, 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 $\xi_0\xi_1\xi_2..\xi_i..\xi_n\xi_{n+1}$ is written left-justified on the otherwise blank tape, we will represent it by a configuration of T_n of the form

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

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

> *"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, $i = 0, 1, .., n+1$:

$$S_{i,q} R_{i,\xi} \rightarrow_A S_{i_D,q'} R_{i,\eta}, \qquad (2)$$

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 $x_1x_2..x_n$ 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 $x_1x_2..x_n$, 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 $x_1x_2..x_n$ 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:

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

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 $\mathtt{value}_v(W_n)$ 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 terminal vertex 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 $\mathtt{value}_v(W_n)$ to the *accepting* configuration represented by Z_n.

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

Comment 6.1 The signature of T_n consists only of $O(n)$ 0-ary predicate symbols. The number of the constants invoked is zero.

Comment 6.2 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 Corollary 6.4.

Nevertheless, we get `PSPACE`-hardness also in 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 $\xi_0\xi_1\xi_2..\xi_i..\xi_n\xi_{n+1}$ is written left-justified on the otherwise blank tape, now it is represented as (cf. (1)):

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

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

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

The *faithfulness* of the modified encoding is established by

Lemma 6.2 *Given an input string $x_1x_2..x_n$ 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 $x_1x_2..x_n$, 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 $x_1x_2..x_n$ 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. ■

Corollary 6.1 *The planning problem remains* PSPACE*-hard even for local state transition systems where a finite set of (unary) predicate symbols is fixed in advance, but there is no limitation for their finite domains.*

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

Corollary 6.3 *The collaborative planning problem with privacy is* PSPACE*-hard.*

Proof. The hardness is already obtained in the single agent case where the privacy condition is vacuously true. ■

6.2. Upper Bounds

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 (including 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:*

$$L_T(m) \leq \binom{m+S_T}{m} = \frac{(m+S_T)!}{m! \cdot (S_T)!} \leq$$

$$\leq \min\{(S_T+1)^m, (m+1)^{S_T}\}.$$

Corollary 6.4 (PTIME for a fixed signature) *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 pair of inputs:

(i) *to any local state transition system T with signature Σ such that each of the actions in R_T is non-lengthening, and R_T is decidable in* `polytime` *by a program ρ,*

(ii) *and to any configurations W and Z,*

which:

(a) *determines whether there is a plan based on R_T, leading from W to the partial goal Z, and*

(b) *if the answer is positive, makes such a plan (of the minimal length), and*

(c) *runs in* `polynomial time` *with respect to the two parameters:*

  (c1) *the binary length of a description of the program ρ*

  (c2) *and m, the least upper bound of the number of facts in each of W and Z.*

Proof. Since the number of configurations is bounded by $L_T(m)$, the number of *closed* actions is bounded by $L_T(m)^2$. The length of any successful computation is easily seen to be at most $L_T(m)$ since we should never have to repeat a configuration during a computation. We can generate each axiom in $\mathtt{TIME}_\rho(L_T(m))$, which is a polynomial in $L_T(m)$. Putting these together, the total time is $O(\mathtt{TIME}_\rho(L_T(m)) \times L_T(m)^2)$. By Lemma 6.3, $L_T(m) \leq (m+1)^{S_T}$. Since we have a fixed finite signature, S_T is a fixed constant. Thus the total time necessary is polynomial. ■

Corollary 6.5 *For a fixed signature, the collaborative planning problem with privacy can be solved in polynomial time.*

Proof. By Corollary 6.4, we can determine the existence of a plan leading to the partial goal Z in polynomial time. We must also determine the non-existence of a plan leading to each fact of the form $Q'(t,\bar{u})$ or $Q_B(t,\bar{u})$, whose presence would violate the security condition. There are a fixed number of such facts (bounded by S_T), and each can be checked in polynomial time. ■

The next theorem shows a PSPACE upper bound for deciding the existence of a plan which leads to any of a finite set of partial goals. It would be sufficient to consider a single partial goal to determine the existence of a plan, but this more general case will be useful later when deciding whether or not the privacy condition is violated.

Theorem 6.3 (PSPACE) *There is an algorithm α' applicable to the following pair of inputs:*

(i) *any local state transition system T over a finite signature such that each of the actions in R_T is non-lengthening, and R_T is decidable by a program ρ in* `pspace`*, and*

(ii) *any configurations $W, Z_1, .., Z_k$,*

which:

(a) *determines whether there is a plan based on R_T leading from W to any of the partial goals $Z_1, .., Z_k$,*

(b) *and runs in* `polynomial space` *with respect to the two parameters:*

  (b1) *the binary length of a description of the program ρ*

  (b2) *and $m \cdot \log_2 S_T$, where m is the least upper bound of the number of facts in each of $W, Z_1, .., Z_k$.*

(It presupposes that $k \leq L_T(m)$.)

Proof. The main ideas are as follows.

First, we show that the *decision problem* under consideration belongs to `PSPACE` by means of the following `npspace` construction.

Let $\mathcal{Z}_0$ be the set which consists of all configurations H of the form $Z_j V$, where $1 \leq j \leq k$ and V is a state such that the number of facts of $Z_j V$ is bounded by m. This is the set of goals with at most m facts whose *exact* reachability implies the reachability of some partial goal Z_j.

We begin with $W_0 := W$.

For any t, if W_t is not yet a member of $\mathcal{Z}_0$, then we obtain W_{t+1} by guessing an action $XX' \rightarrow_A YY'$ which is applicable in configuration W_t. Then W_{t+1} is the configuration which results from applying the action to W_t, and we replace the old W_t by the new W_{t+1}.

Let us now determine the space required for each t-th step. We need to record the configuration W_t. Since the total number of configurations is bounded by $L_T(m)$, we can do this in $\log_2(L_T(m))$ space. By Lemma 6.3, we get the following inequalities:

$$\log_2 L_T(m) \leq \log_2(S_T + 1)^m \leq 2m \log_2(S_T)$$

Similarly, the action which is guessed can be stored on $O(m \log_2(S_T))$ space. Each axiom is generated by the program ρ in $\mathtt{SPACE}_\rho(O(m \log_2(S_T)))$, which is polynomial in $m \log_2(S_T)$, since ρ runs in polynomial space. Thus the total space required for each step is polynomial in $m \log_2(S_T)$ and the binary length of the description of ρ.

We can show that $W \rightsquigarrow^*_{R_T} Z_j$ for some $1 \leq j \leq k$ iff there is a sequence of at most $L_T(m)$ guesses that ends in $\mathcal{Z}_0$.

The technique of Savitch's theorem [12] shows how to perform this procedure in `pspace`.

To generate the plan that is hidden inside this construction, we modify the construction so that, each $XX' \rightarrow_A YY'$ from the plan will be eventually kept on additional $O(m \cdot \log_2 S_T)$ space. ■

As a corollary, we get that the collaborative planning problem with privacy is in `PSPACE`.

Corollary 6.6 *There is an algorithm α' applicable to the following triple of inputs:*

(i) *any local state transition system T over a finite signature such that each of the actions in R_T is non-lengthening, and R_T is decidable by a program ρ in* `pspace`*, and*

(ii) *any configurations W and Z, and*

(iii) *any finite list t_i^A of terms owned by A (to be protected from others).*

which:

(a) *determines whether, for each A, t_i^A are protected from all other Bs,*

(b) *determines whether there is a plan based on R_T leading from W to the partial goal Z,*

(c) *and runs in* `polynomial space` *with respect to the two parameters:*

  (c1) *the binary length of a description of the program ρ*

  (c2) *and $m \cdot \log_2 S_T$, where m is the least upper bound of the number of facts in each of W and Z,*

Proof. By Theorem 6.3, part (b) can be checked in polynomial space. Since `PSPACE` = `COPSPACE`, we can check part (a) for each list $t_1^A, .., t_k^A$ in polynomial space (again by Theorem 6.3). ■

7. Related Work

The formalism presented in this paper has some similarities to other work. The Multiset Rewriting formalism (MSR) presented in [5] and [3] 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 [5] 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.

The work in [7] and [6] uses similar techniques to provide an abstract model for distributed access control. While access control and our privacy concerns share some similarities, our focus is on the interaction between the release and protection of resources in a general collaborative setting. Nevertheless, it would be interesting to investigate a more precise relationship between our approach and the use of linear logic in [7] and [6].

The work in [24] presents rewriting systems by means of conditional rewriting logic. This logic is proposed as a general model for concurrent systems. It presents Maude, a programming language which helps to unify concurrent programming with functional and object-oriented programming.

Many models of information flow provide an explicit mechanism for intentionally downgrading or declassifying information. The work in [31] and [25] addresses non-interference in such a setting. They consider the possibility of abusing explicit declassification mechanisms to provide for the unintentional leak of information.

There has been work on specific forms of collaboration. For example [13] and [29] consider scientific data sharing, and [30] considers the controlled release of medical information. These approaches are not as abstract as ours and they focus more on the implementation of local policies.

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' privacy/secrecy of information or resources. 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 which protects the privacy/secrecy of all participants is PSPACE-complete. We saw that by fixing in advance the number of constants and predicates, the collaborative planning problem with privacy/secrecy is solvable in polynomial time.

Currently, we have considered local state transition systems with constants and predicates. In the future we would like to consider a richer language of functional terms and appropriate ways of bounding their complexity. We would also like to consider the case with existential quantifiers. This would allow the participants to create new values as in [5]. We would like to investigate the use of the linear logic connective $\oplus$ to model branching nondeterminism resulting from actions with nondeterministic effects. We also believe our model provides the ability to trace the leak of information back to its point of origin. This would provide a sort of auditing mechanism. We would like to investigate the relationship to non-interference and information flow in this collaborative setting. In addition, we have assumed a threat model which isolates the participants from outside intruders. Adjusting this model to consider a more general setting is desirable.

While we noted in Section 4.1 that our formalism is able to distinguish between obsolete secrets and current secrets, we would like to complete a more thorough investigation of how this may be used. In addition, our system currently only distinguishes between private (High) and public (Low). In practice one might use a more complicated structure such as a partially ordered set. We would also like to explore the use of some sort of utility functions which weigh the relative importance of information release and protection. They can guide the agents' decisions on whether or not to release information. This could lead us to a more fine-grained definition of the protection of privacy.

Acknowledgements

Thanks to Zack Ives, Pat Lincoln, Ralph Wachter and Steve Zdancewic and other members of the Penn Computer Security Seminar for their useful suggestions and comments.

References

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

[2] I. Cervesato, N. Durgin, M. Kanovich, and A. Scedrov. Interpreting strands in linear logic. In H. Veith, N. Heintze, and E. Clark, editors, *2000 Workshop on Formal Methods and Computer Security –FMCS'00*, Chicago, IL, 2000.

[3] I. Cervesato and A. Scedrov. Relating state-based and process-based concurrency through linear logic. In R. de Queiroz, editor, *Thirteenth Workshop on Logic, Language, Information and Computation — WoLLIC'06*, pages 145–176, Stanford, CA, 18–21 July 2006. Elsevier ENTCS 165.

[4] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–377, 1987.

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

[6] D. Garg, L. Bauer, K. D. Bowers, F. Pfenning, and M. K. Reiter. A linear logic of authorization and knowledge. In *Proceedings of the 11th European Symposium on Research in Computer Science (ESORICS'06)*, volume 4189 of *Springer Lecture Notes in Computer Science*, pages 297–312. Springer-Verlag, 2006.

[7] D. Garg and F. Pfenning. Non-interference in constructive authorization logic. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW)*, pages 283–296, 2006.

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

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

[10] J.-Y. Girard. Linear logic: Its syntax and semantics. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecutre Notes*, pages 1–42. Cambridge University Press, 1995.

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

[12] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[13] Z. G. Ives, N. Khandelwal, A. Kapur, and M. Cakir. ORCHESTRA: Rapid, collaborative sharing of dynamic data. In *Conference on Innovative Data Systems Research (CIDR)*, pages 107–118, 2005.

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

[15] M. I. Kanovich. Horn programming in linear logic is NP-complete. In *Proc. 7-th Annual IEEE Syposium on Logic in Computer Science, Santa Cruz*, pages 200–210, 1992.

[16] M. I. Kanovich. The complexity of Horn fragments of linear logic. *Annals of Pure and Applied Logic*, 69:195–241, 1994.

[17] M. I. Kanovich. Linear logic as a logic of computations. *Annals of Pure and Applied Logic*, 67:183–212, 1994.

[18] M. I. Kanovich. The direct simulation of Minsky machines in linear logic. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Notes*, pages 123–145, 1995.

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

[20] M. Masseron, C. Tollu, and J. Vauzeilles. Generating plans in linear logic I: Actions as proofs. *Theoretical Computer Science*, 113(2):349–370, 1993.

[21] E. W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM J. Comput.*, 13(3):441–460, 1984.

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

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

[24] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. In *Selected papers of the Second Workshop on Concurrency and compositionality*, pages 73–155, Essex, UK, 1992. Elsevier Science Publishers Ltd.

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

[26] N. J. Nilsson. *Principles of Artificial Intelligence*. Springer, Berlin, 1980.

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

[28] A. Scedrov. Linear logic and computation: A survey. In H. Schwichtenberg, editor, *Proof and Computation, Proceedings Marktoberdorf Summer School 1993*, pages 281–298. NATO Advanced Science Institutes, Series F, Springer-Verlag, Berlin, 1994.

[29] N. E. Taylor and Z. G. Ives. Reconciling while tolerating disagreement in collaborative data sharing. In *ACM SIGMOD Conference on Management of Data*, pages 13–24, 2006.

[30] G. Wiederhold, M. Bilello, V. Sarathy, and X. Qian. Protecting collaboration. In *Proc. 19th NIST-NCSC National Information Systems Security Conference*, pages 561–569, 1996.

[31] S. Zdancewic and A. C. Myers. Robust declassification. In *Proc. 14th IEEE Computer Securtiy Foundations Workshop (CSFW)*, pages 15–23, 2001.

A. Some Rules of Affine Logic

Structural rules:

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

Cut rules

$$\frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2, A \vdash C}{\Gamma; \Delta_1, \Delta_2 \vdash C}\,\mathbf{cut}$$

$$\frac{\Gamma; \cdot \vdash A \quad \Gamma, A; \Delta \vdash C}{\Gamma; \Delta \vdash C}\,\mathbf{cut!}$$

Left rules

$$\frac{\Gamma; \Delta \vdash C}{\Gamma; \Delta, \mathbf{1} \vdash C}\,\mathbf{1}\text{-}\ell \qquad \frac{\Gamma; \Delta, A_1, A_2 \vdash C}{\Gamma; \Delta, A_1 \otimes A_2 \vdash C}\otimes\text{-}\ell$$

$$\frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2, B \vdash C}{\Gamma; \Delta_1, \Delta_2, A \multimap B \vdash C}\multimap\text{-}\ell$$

$$\frac{\Gamma; \Delta, A \vdash C}{\Gamma; \Delta, \exists x.A \vdash C}\,\exists\text{-}\ell$$

Right rules

$$\frac{}{\Gamma; \cdot \vdash \mathbf{1}}\,\mathbf{1}\text{-}r \qquad \frac{\Gamma; \Delta_1 \vdash C_1 \quad \Gamma; \Delta_2 \vdash C_2}{\Gamma; \Delta_1, \Delta_2 \vdash C_1 \otimes C_2}\otimes\text{-}r$$

$$\frac{\Gamma; \Delta \vdash [t/x]C}{\Gamma; \Delta \vdash \exists x.C}\,\exists\text{-}r$$