RC23996 (W0607-005) July 5, 2006
Computer Science

IBM Research Report

Automatic Composition of Secure Workflows

Marc Lelarge, Zhen Liu, Anton Riabov
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Automatic Composition of Secure Workflows

Marc Lelarge, Zhen Liu and Anton Riabov
IBM T.J. Watson Research Center
P.O. Box 704, Yorktown Heights, New York, 10598
marc.lelarge@ens.fr, zhenl@us.ibm.com, riabov@us.ibm.com

Abstract

Automatic goal-driven composition of information processing workflows, or workflow planning, has become an active area of research in recent years. Various workflow planning methods have been proposed for automatic application development in systems like Web services, stream processing and grid computing based on compositional architectures. Significant progress has been made on the development of composition methods and on the definition of composition rules. The composition rules can be specified based on the schema, interface and semantics-driven compatibility of processes and data. More importantly, in many practical applications the workflows must be executed under access control policies. In this paper we introduce and study the problem of workflow planning under the constraints of MLS and the Bell-LaPadula model. This problem arises in the context of our implementation of a large-scale stream processing system that can process a wide variety of different inquiries submitted by end users. Extending well-known results from AI planning literature, we first show that under certain simplifying assumptions the workflows satisfying Bell-LaPadula model constraints can be constructed in linear time. Further we show that the problem becomes NP-complete once the use of trusted downgraders for data declassification is allowed. Next, we identify a number of special conditions under which the workflows can still be constructed in polynomial time, even when the use of downgraders is allowed. Finally, we analyze the impact of Chinese Wall constraints on the complexity of the composition problem, and describe an efficient algorithm for composing workflows under these constraints. The proposed approach can be used with any lattice-based access control policies, including Biba integrity model.

1 Introduction

Automatic composition of information processing workflows, or workflow planning, has become an active area of research in recent years. Growing need of reliable and efficeint application development together with the progress in modular software design and reusable software component frameworks have fueled the interest of researchers in this area. Automatic workflow planning tools can give end users the power to interconnect the components and create new processing workflows for their tasks on demand, based solely on the specification of requirements. Workflow planning systems such as Pegasus [GDB+04] and Sekitei [KIK03] can ensure that the workflows use available resources as efficiently as possible. Furthermore, the systems that implement automatic workflow planning are more resilient than those that rely on manual composition: the automatic planning systems can easily adapt to changes by replanning existing workflows under the changed conditions.

The literature on workflow planning is extensive and spans diverse application areas where compositional architectures are used, including Web services [DGAV04, KS03, PTB05], grid computing [BDG+03, GDB+04], and stream processing [RL05]. CHAMPS project [BKH05] is an example of successful practical use of planning techniques for composition of change management workflows.

In previous work on planning, however, the practically important issue of access control constraints on the automatically constructed workflows has not received enough attention. In various business applications security policies place restrictions on the information flow. These restrictions naturally constrain

the set of valid workflows that can be executed in the system. The workflow planning methods designed for these environment, in addition to resource, quality, semantic and other constraints, must support access control constraints imposed by the security policies.

In this paper we focus on access constraints defined by lattice-based access control models [San93]. One example of lattice-based access control is Bell-LaPadula policy [BL76] commonly implemented in MLS (multi-level secure) systems. MLS systems can process and store information at a variety of different sensitivity levels without disclosing sensitive information to an unauthorized entity. The MLS model is based on the following principle: 1) clearance is determined for each user; 2) information is marked with a sensitivity label; and 3) access control decisions are made based upon the clearance of the user and the sensitivity of the information. Bell-LaPadula policy requires that no high level information should be allowed to pass to lower level users/processes and that lower level information should be available to higher level users/processes [BL76]. The latter restriction is often considered too rigid, and can be relaxed. In that case the use of trusted components called downgraders are allowed for declassification of data objects. We discuss the downgraders and associated planning complexity issues in Section 3.4.

The implementation of automatic workflow planning techniques in MLS or other systems with access controls can result in significant benefits. These techniques are used in many fields to improve efficiency, ease of use and reduce the time of reaction to changes. In addition to these benefits, in systems where the use of downgraders is possible and allowed by policy, workflow planning has an important additional advantage. Without automatic planning, the users with limited access rights would not be able to construct workflows that include downgraders, since these users do not have access to the components that precede the downgraders in the workflow. The workflow planner, however, can be implemented as a trusted component, and can construct workflows that include any number of sensitive components and sources, as long as the output of the workflow is accessible for the users who initiated the planning request. In this case the users may not be allowed to see the complete structure of the created workflow, but the workflow can still be automatically executed and deliver the results to the users in full compliance with security policy.

1.1 System Architecture Assumptions

Our work on automatic planning is carried out in the framework of a new large-scale distributed stream processing system, that we refer to as System S, which allows the automatic deployment of automatically built stream processing plans in the distributed system. In order to describe the role of planning in MLS access control more generally, we will now outline a general system architecture within which our approach can be applied. Our target system consists of multiple machines connected to a network. Each of these machines can host one or more software components. The functionality of the components is not restricted in any way, and can range from database management to computation; however, the components must expose and describe all input and output data interfaces in a format that is recognized by a common compositional framework.

The compositional framework provides all functionality necessary to establish links between the input and output ports of components. When an output port of one component is linked to an input port of another, all data produced at the output port of the first component will arrive at the input port of the second component. One output port may be linked to multiple input ports, and in that case each input port will receive a copy of the data sent via the output port. The communication can take place both between the components running on the same machine, as well as on different machines. Components can also be linked to external sources or send data to external recipients.

To remain consistent with the existing literature on the subject, we will refer to the composition of interlinked components as *workflow*, or *composite application*. The procedure of workflow composition is commonly referred to as *planning*.

In the context of this system, the Bell-LaPadula model with downgraders require the assignment read and write labels to each of the components. Access control policy states that the read label of the component should be higher or equal to the label of each incoming stream; similarly, the label of each output

stream should be the same or higher than the write label. The write labels are permitted to be lower than the read labels only for the trusted components, i.e. downgraders.

In our architecture we make the distinction between access control in run time and during planning. The workflows can be composed of already deployed components, or using abstract component descriptors with subsequent instantiation and deployment to the machines. However, in both of these cases run-time information assurance infrastructure, including secure communication channels, secure operating systems, prevention of covert channels, etc., will be necessary to prevent unauthorized access. We will assume that such an infrastructure is in place, and focus our attention on the assignment of the labels to streams and components and on access policy compliance verification during workflow planning.

1.2 Contribution and Paper Organization

The approaches proposed for automatic workflow composition, including [BDG+03, DGAV04, GDB+04, KIK03, KS03], use the techniques developed in the area of AI planning. Planning literature describes methods for the composition of series of operations that transform the world of the problem domain from the initial state to the goal state (see survey [RH01]).

In this paper we show that planning techniques can be used to compose workflows that are compliant with MLS security policy constraints. The basic planning framework compliant with Bell-LaPadula (BLP) model can be extended to perform automatic workflow composition in systems with other lattice-based models, with trusted downgraders and with Chinese Wall policy. The workflows satisfying the BLP constraints can be constructed by the planner very efficiently. The number of operations performed by our algorithm grows linearly in the number of components. And, although the general problem of planning BLP-compliant workflows with the inclusion of downgraders is NP-complete in the worst case, efficient solutions can be given if the system configuration and the security policy have certain structural properties.

The structure of the paper is the following: in Section 2, we formally define the workflow planning problem. In Section 3, we extend the model to include the constraints of the BLP model. In particular, if trusted downgraders for data declassification are allowed, the problem becomes much harder (NP-complete). Further in Section 4 we describe several special conditions that enable efficient planning of secure workflows with downgraders. In Section 5 we show how the constraints of the Chinese Wall policy can be added to our planning framework, and provide an efficient solution algorithm. Concentrating on the presentation of the main results, we defer the technical proofs to the Appendices A, B, C and D.

2 Basic Workflow Planning Model

In this section we will formally define the elements of the workflow planning problem studied in this paper. The expressivity of the proposed model is limited so that only the workflow composition constraints that are independent of the context can be modeled. This basic model yields a simple solution: as we show below, a feasible workflow (if one exists) can be constructed in time polynomial in the size of the problem. Despite the seeming over-simplicity and even impracticality of the model, it captures a large family of composition constraints arising in practice. More importantly, this model allows us to make a clear distinction between the class of easily solvable planning problems in the absence of access controls, and the corresponding class of NP-complete problems under access control constraints.

2.1 Modeling Considerations and Limitations

Before describing our model in detail we will briefly review the previous work on modeling. Many alternative models have been proposed for this problem in the literature, concentrating on various aspects of this complex problem. For example, numerical resource constraints were studied in [KIK03]. Furthermore, to create meaningful workflows in many scenarios that arise in practice it is important to ensure that the planner can understand the semantics of workflows in the context of the application domain. The efforts

in this direction have resulted in a standard called OWL-S [MPM$^+$04] (formerly DAML-S) for semantic descriptions of the capabilities of web services via ontologies. One of the goals of OWL-S standard is to provide the support for automatic semantics-based composition of web services. Initial investigation of planning algorithms and planning models that can take into account the semantic constraints are described, for example, in [WPS$^+$03] and [AVMM04]. The detailed process models for modeling individual services in Web Services domain are considered in [PTB05].

In a general model of data flow, the workflows are composed of components which are also referred to as operators or actions. Each operator can receive input from other operators or from the primal sources, i.e. from the sources of data that are available before any parts of the workflow are executed. The workflow composition constraints discussed in the literature can be divided into two mutually exclusive categories: context-dependent constraints and context-independent constraints. Under the context-independent constraints model each operator is associated with 1) the descriptions of the input data that are required for applying the operator; and 2) the descriptions of the output data that are produced by the operator. The context-dependent constraints are the restrictions on workflows that are not solely defined by the data compatibility between the data and input requirements of an operator. For example, if the description of output data produced by an operator significantly changes when the operator is applied to different sets of input data, therefore causing changes in the set of operators that can be applied further, the context-dependent model must be used. Many numerical constraints, such as resource constraints limiting total resource usage for a workflow, also fall within the category of context-dependent constraints.

The basic model described below supports only the context-independent constraints. Constraints of this type are very common in practice, and appear in virtually every workflow planning scenario described in the literature. Various constraints originating from the requirements on semantic compatibility or format compatibility between the inputs and the outputs of the components belong to the context-independent category. Further in the paper we will use this basic model to study the additional computational complexity associated with the introduction of security constraints, as well as the methods for reducing that complexity. We note here that security constraints that we will add to this basic model in Section 3 are a form of context-dependent constraints.

2.2 Formulation of the Basic Workflow Planning Model

To describe the context-independent constraints we will use the concept of data type for expressing the compatibility between the inputs and the outputs of the operators. In particular, we will assume that all data flowing through the system are described by a type or by a set of types. We will further assume that there are a total of n distinct data types used in formulating the problem. In the initial state, before any operators are included into the workflow (i.e, applied), only those types that are produced by the primal sources are available. Each operator has a set of types that must be available at the time when the operator is applied, i.e., the precondition of the operator, and a set of new types that become available after the operator is applied, i.e., the effect of the operator. The workflow planning problem then is to construct a sequence (or a partial ordering) of operators, application of which in that order will make available all types that are included into a given goal set of types.

Throughout the paper we will follow the notation defined below. Let vector $x \in \{0,1\}^n$ be the state vector, describing the set of currently available types. Each non-zero component, i.e. $x_j \neq 0$, of this vector corresponds to a currently available type. For any two vectors x and y, the ordering $x \leq y$ is interpreted componentwise and is thus equivalent to $x_j \leq y_j$ for all $1 \leq j \leq n$.

Each operator $\theta \in O$ is described by a pair of the precondition $p(\theta) \in \{0,1\}^n$ and the effect $a(\theta) \in \{0,1\}^n$, i.e. $\theta := \langle p(\theta), a(\theta) \rangle$, with vector components denoted by $p_i(\theta)$ and $a_i(\theta)$. Let $m := |O|$ be the number of operators. When the operator θ is applied in state x, it causes the transition to state $\theta(x)$:

$$\theta(x) := \begin{cases} \max(x, a(\theta)), & \text{if } x \geq p(\theta); \\ x, & \text{otherwise;} \end{cases} \tag{1}$$

According to the definition of the state transition above, if the precondition of an operator θ is not satisfied, i.e. if $x \not\geq p(\theta)$, the operator can still be applied (with our way of writing). However, applying such an operator will not change the state.

A plan π is defined as a sequence of operators. If $\pi = (\theta_1, \ldots, \theta_k)$, we will say that the length of the plan π is k. The effect of the plan π on state x is expressed by the following composition:

$$\pi(x) \quad := \quad \theta_k \cdot \theta_{k-1} \cdot \cdots \cdot \theta_1(x),$$

where we first apply the operator θ_1, then θ_2 and so on.

Given two plans $\alpha = (a_1, \ldots, a_k)$ and $\beta = (b_1, \ldots, b_l)$, we denote by $\pi = \alpha \odot \beta$ the plan obtained by concatenation of the two plans, namely $\pi = (a_1, \ldots, a_k, b_1, \ldots, b_l)$. We define the concatenation of a plan with a set of operators similarly, assuming that the operators in the set are applied in an arbitrary order. Note that with this notation for all operators θ and plans π, $\pi \odot \theta(x) = \theta \cdot \pi(x)$.

Given an initial condition $x^0 \in \{0,1\}^n$ and a goal $g \in \{0,1\}^n$, we will say that the plan π achieves the goal g with the initial condition x^0 if $\pi(x^0) \geq g$. We will say that π solves (or is a solution of) the planning problem. We denote the set of all planning problem solutions by $\mathcal{P}(x^0, g)$, i.e., $\pi \in \mathcal{P}(x^0, g)$.

The planning problem described above is equivalent to the propositional STRIPS problem with empty delete lists. STRIPS is a well-known planning domain model described, for example, in [FHN72]. Next, we will make several additional assumptions about the data structures that will be used by the planning algorithm. These assumptions do not significantly change the model, but are helpful in complexity analysis of algorithms. Figure 1 below shows a graphical representation of an operator.

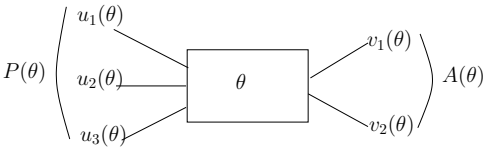

Figure 1: Operator

We will assume that for all operators the maximum number of nonzeros in the precondition and effect vectors is bounded. Namely, all operators are stored in a table of size m, where the i-th entry corresponds to the operator θ_i and points to two lists of variables: $P(\theta_i) := \{u \mid p_u(\theta_i) = 1\}$ corresponds to the operator precondition and $A(\theta_i) := \{v \mid a_v(\theta_i) = 1\}$ corresponds to the operator effect. We assume that $|P(\theta)| \leq C$ and $|A(\theta)| \leq C$ for all operators $\theta \in O$ and for some constant $C \geq 1$, which in particular is independent of n and m.

Without loss of generality the following inequality can be assumed to hold in all problem instances:

$$n \leq 2mC. \tag{2}$$

This bound on n is derived from the fact that there are at most $2C$ nonzeros in the precondition and effect vectors of each operator, and therefore at most $2C$ types are listed in operator description. If in some formulation inequality (2) is violated, the types that are not used in any of the operator descriptions can be removed without changing the problem, and n can be reduced.

For convenience we introduce the following notation. For any $x \in \{0,1\}^n$, let

$$\mathcal{T}(x) := \{i \mid x_i = 1\} \subseteq \{1, 2, \ldots, n\},$$

be the set of types available in state x, defined by the set of non-zeros in the vector x.

Figure 2 shows a graphical representation of a plan. The black dots on the left side represent the data types available in the initial state, $\mathcal{T}(x^0)$, and the black boxes on the right side represent the goal set of data types, $\mathcal{T}(g)$. The links between the operator rectangles corresponds to the types required (on the left of the

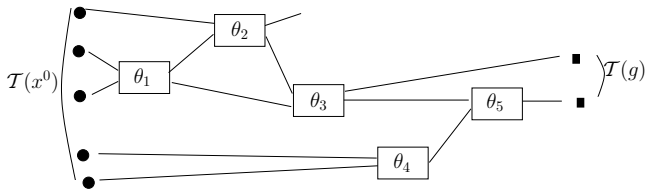

Figure 2: Plan $\pi = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5) \in \mathcal{P}(x^0, g)$.

operator) and produced (on the right) by the operator. The figure shows that the plan $\pi = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ is in $\mathcal{P}(x^0, g)$. Note that the plan $(\theta_4, \theta_1, \theta_2, \theta_3, \theta_5)$ is also a feasible solution, and therefore is in $\mathcal{P}(x^0, g)$.

It is known that STRIPS planning problem without deletion can be solved in polynomial time [Byl94]. In fact, planners can solve this problem in time linear in n and m. The following proposition states this result in our framework and describes a basic linear time planning algorithm.

Proposition 1 *Given a set of m operators satisfying previous conditions, for any initial condition x^0 and a goal g, in $\mathcal{O}(m)$ operations an element of $\mathcal{P}(x^0, g)$ can be found if $\mathcal{P}(x^0, g) \neq \emptyset$. If such an element is not found by this procedure, then necessarily $\mathcal{P}(x^0, g) = \emptyset$.*

Below we define the algorithm **FAST_GREEDY** that finds a solution in linear time, or proves that no solutions exist. While similar algorithms have been designed for STRIPS planners before, we describe this particular implementation because it will be useful in future analysis of more complex formulations. We include the complete complexity analysis of the algorithm in Appendix A. In this appendix we also identify the sufficient conditions on the operators for this proposition (see Lemma 1 in the Appendix). These conditions, in addition to being of interest in themselves, will simplify the analysis of the planning problem with security constraints discussed in the next section.

```
    FAST_GREEDY(x) :
1   π := ∅;
2   W := T(x); D := 0;
3   while W ≠ ∅ do
4     take (and delete) i from W;
5     D := D + e^i;
6     for each operator θ ∈ O such that p_i(θ) = 1 do
7       p_i(θ) := 0;
8       if p(θ) = 0 then do
            W := W ∪ {i, a_i(θ) − D_i = 1};
            π := π ⊙ θ; od
9     od
10  od
11  Return  π;
```

Above, for all $1 \leq i \leq n$, $e^i \in \{0,1\}^n$ is an i-th unit vector, defined as $e^i_j = 0$ for $j \neq i$ and $e^i_i = 1$.

In what follows we will denote $\mathbf{FG}(x^0) = \pi_{FG}(x^0)$ where π_{FG} is the plan returned by the algorithm with initial condition x^0, $\pi_{FG} = \mathbf{FAST\_GREEDY}(x^0)$. The set of planning problem solutions $\mathcal{P}(x^0, g)$ is not empty if and only if $\mathbf{FG}(x^0) \geq g$ and in this case, $\mathbf{FAST\_GREEDY}(x^0)$ is an element of this set. As shown by Proposition 1, solving the basic planning problem is easy, thanks to the monotonicity property: as long as possible we can produce new types and stop when all operators have been

applied. Note that the problem of finding an optimal plan (i.e. a plan with minimal length) in $\mathcal{P}(x^0, g)$ is a much harder task. It has been shown that it is an NP-hard problem (see [Byl94]). We will not deal with this optimality criterium here since the main purpose of our work is first to show that the introduction of the security constraints increases the complexity of the basic planning problem and second to give conditions that enable efficient efficient planning of secure workflows.

3 Workflow Planning with Security Constraints

In this section we add the constraints of the Bell-LaPadula security model to our basic workflow planning model, and describe an efficient algorithm for planning with these constraints. We further show that if the use of trusted downgraders is allowed, finding a feasible plan under security constraints is an NP-complete problem. Due to our general approach to modeling the constraints using a label lattice, the results of this section are applicable not only to the Bell-LaPadula model, but to other lattice-based models, such as the Biba integrity model [Bib77, San93].

3.1 The Bell-LaPadula Model

First, we briefly review the basics of the Bell-LaPadula (BLP) model [BL76]. BLP is a lattice-based access control model for information flows [San93, Den76]. A lattice $(\mathbb{L}, \prec)$ is a partially ordered set with dominance relation $\prec$, in which an upper bound and a lower bound are defined for each pair of elements. Let $a \vee b$ (resp. $a \wedge b$) denote the upper (respectively, lower) bound of a and b. Under the BLP model, all users (i.e., subjects) and all data (i.e., objects) are assigned security labels, which are elements of the lattice. A particular user is allowed access only to the data with labels that are dominated by the label (i.e., access class) of that user.

The following is a classic example:

- Let $\mathcal{H}$ be a totally ordered set of sensitivity levels:

$$\mathcal{H} = \{Public \leq_{\mathcal{H}} Classified \leq_{\mathcal{H}} Secret \leq_{\mathcal{H}} Top\text{-}Secret\}.$$

- Let $\mathcal{C}$ be a set of categories: $\mathcal{C} = \{PERSONNEL, ENGINEERING\}$.
- A security label is a pair $\ell = (h, c)$ with $h \in \mathcal{H}$ and $c \subset \mathcal{C}$.

The dominance relation $\prec$ is defined as a componentwise order, i.e.

$$\ell_1 = (h_1, c_1) \prec \ell_2 = (h_2, c_2) \quad \text{if and only if} \quad h_1 \leq_{\mathcal{H}} h_2 \text{ and } c_1 \subseteq c_2.$$

For example, $(Pub., \{PERS.\}) \prec (Sec., \{PERS., ENG.\})$, but $(Pub., \{PERS.\}) \not\prec (Pub., \{ENG.\})$. The upper and the lower bound in this example are defined as follows:

$$\begin{aligned} \ell_1 \wedge \ell_2 &:= (\min\{h_1, h_2\}, c_1 \cap c_2) \\ \ell_1 \vee \ell_2 &:= (\max\{h_1, h_2\}, c_1 \cup c_2) \end{aligned}$$

The BLP model defines two rules for making access control decisions based on the object label of the data and the access class label of the user. The rules, also called properties, are defined as follows:

- The simple security (ss-)property of the BLP model allows read access to the data, only if the access class label of the user dominates the correspond object label (no read up rule).
- The star (*-)property of the BLP model allows write access to the data only if the corresponding object label dominates the access class label of the user (no write down rule).

3.2 Workflow Planning Model with Bell-LaPadula Security Constraints

Our objective in representing the BLP constraints in the workflow planning model is to define a problem of composing a workflow such that the output, i.e. the goal types, of the workflow can be accessed by a specified access class. Data coming into the system from primal sources, i.e. the types of the initial state, can be labeled with arbitrary object labels, which are provided as input to the planner. In the process of planning the planner will also assign subject labels to the operators processing the data, as well as object labels to all produced types, and do that such that the BLP policy is satisfied over the entire workflow.

The objects in the basic workflow planning model correspond to the data types, and therefore we extend the model by assigning a security label variable to each data type. As a result, the state space is extended from $\{0,1\}^n$ in the basic formulation to a subset $\mathcal{S}$ of $\{0,1\}^n \times \mathbb{L}^n$, where $(\mathbb{L}, \prec)$ is a lattice corresponding to the set of security labels. We will denote the state by a pair (x, ℓ). With each type $i \in \{1, \ldots, n\}$ represented by the zero-one indicator x_i, we associate a label $\ell_i \in \mathbb{L}$. If type i is not available in state (x, ℓ), i.e. if $x_i = 0$, then the corresponding label ℓ_i is set to a default value. More precisely, we define the set $\mathcal{S}$ as follows

$$\mathcal{S} = \{(x, \ell) \in \{0,1\}^n \times \mathbb{L}^n, \text{ if } x_i = 0 \text{ then } \ell_i = \top\}, \tag{3}$$

where $\top \in \mathbb{L}$ is the top element of the lattice, defined as following: for any $\ell \in \mathbb{L}$, the following holds: $\ell \prec \top$, $\ell \wedge \top = \ell$ and $\ell \vee \top = \top$.

In the extended workflow planning model the operators act on the pair (x, ℓ) and produce another pair $\theta(x, \ell) = (x', \ell')$. The value of x' is determined using (1) as before. We will discuss the computation of ℓ' separately below. We denote $\underline{\theta}(x, \ell) \equiv \underline{\theta}(x) := x'$, where we suppress the ℓ in $\underline{\theta}(x)$ to stress that x' does not depend on the security label ℓ. Similarly, we denote $\overline{\theta}(x, \ell) := \ell'$.

To enforce the *-property we must ensure that when a new type is made available by the operator θ, the label corresponding to this new type dominates all labels of the input types. In practice the upper bound of the input labels is taken and we have formally:

$$\text{if } [x_j = 0 \text{ and } a_j(\theta) = 1] \quad \text{then} \quad \bigvee_{k,\, p_k(\theta)=1} \ell_k = \overline{\theta}(x, \ell)_j = \ell'_j. \tag{4}$$

Note that an operator produces several data types, all with the same security label.

Equation (4) corresponds to the case when an operator produces a new type j. But it may happen that an operator produces a type that is already in x, namely $x_j = 1$ and $a_j(\theta) = 1$. In this case, the same type j can be obtained with two potentially different security labels, namely ℓ_j (already associated with x_j) and ℓ'_j (defined as in equation (4)). If we have $\ell_j \prec \ell'_j$ (or $\ell'_j \prec \ell_j$), it is clear the lowest label ℓ_j (resp. ℓ'_j) should be chosen as the new value of the label.

In workflow planning this corresponds to the case where the same information has been produced by two different methods and has two different security labels. For example one is Top-Secret because the information has been extracted from a Top-Secret document and the other is Public because the information appears in an unclassified document. In this case, the information should clearly be labeled Public, since a user does not need to have a Top-Secret access class to access the same data. We defer further discussion of related issues to the next section.

However, if we do not assume total order on $\mathbb{L}$ (and we refer to Section 4.1 for the special case of total order), it can happen that none of the two conflicting labels dominates the other. Hence for each type all encountered labels for which domination cannot be determined must be stored. In the example above, we add the new label ℓ'_j to the list:

$$\text{if } x_j = 1 \text{ and, } a_j(\theta) = 1 \text{ then } \overline{\theta}(x, \ell)_j = \left\{\ell'_j, \ell_j\right\},$$

where ℓ'_j is defined as in equation (4). Consequently, if the data type j is used as a precondition of another operator at a later stage, one of the possible labels from that list must be chosen to be used in new

computation (4). Therefore the plan in this extended model must be described as follows:

$$\pi = \left((\theta_1, L_1 = \{\ell_i\}_{i \in P(\theta_1)}), (\theta_2, L_2 = \{\ell_i\}_{i \in P(\theta_2)})\right), \tag{5}$$

where each L_k contains the choice of labels that are used as input labels by the operator θ_k. For example, suppose that at some stage in the plan the same data type i is produced by operators u and v with two different labels ℓ and ℓ' as shown in Figure 3. We must distinguish between the two different plans that can be constructed with the operator θ taking as input (i, ℓ) or (i, ℓ'). This choice is shown by the double-arrow in the figure. Using our definition of a plan, the security labels produced by θ_k are $\bigvee_{i \in L_k} \ell_i$. We denote by $\{\overline{\pi}(x^0, \ell^0)_{i_j}\}$ the set of security labels obtained for type j by plan π. To summarize, plan π produces data types j if $\underline{\pi}(x^0)_j = 1$ and in this case, the plan may produce different copies of this data with respective security labels $\{\overline{\pi}(x^0, \ell^0)_{1_j}, \overline{\pi}(x^0, \ell^0)_{2_j}, \dots\}$.

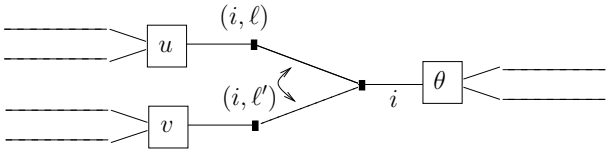

Figure 3: A conflict between labels ℓ and ℓ' for the same type i.

To enforce the ss-property, we must ensure that security level of the user ℓ_{USER} dominates the label of the output data produced by the workflow. The output data corresponds to the data types included in the goal vector g. As we discussed above, each data type may be produced with different security labels and it can be disclosed to the user only if among these security labels, at least one is dominated by ℓ_{USER}. Hence the condition that must be enforced is:

$$\text{if } \underline{\pi}(x^0)_j = g_j = 1 \text{ then, there exists } i_j \text{ such that } \overline{\pi}(x^0, \ell^0)_{i_j} \prec \ell_{USER}. \tag{6}$$

The workflow planning problem is now to find a plan that will achieve the goal g and satisfy equation (6). In other words, the plan π must produce all data types requested in the goal g, i.e. $\pi \in \mathcal{P}(x^0, g)$, and each data type of the result matched to the goal must have a security label less or equal to ℓ_{USER}.

3.3 Planning Algorithm for Workflows with Bell-LaPadula Security Constraints

Despite the seeming complexity of the extended workflow planning model, a minor modification of the planning algorithm for basic model can provide an efficient solution. Note that given Equation (4), the maximum security label of all the types within a plan is the maximum of the input type labels used in the plan. Hence the algorithm to solve the planning problem with security constraints is quite simple. First, we must select all types in x^0 with security labels that are dominated by ℓ_{USER} and remove all other types. Then starting with this new initial state y^0 we must solve the workflow planning problem without the security constraints. In particular, it is no longer required to take into consideration the sets L_k defined in (5) for solving the problem, and the algorithm $\mathbf{FAST\_GREEDY}$ described in the previous section can be used without modification. If $\mathbf{FG}(y^0) \geq g$, then there exists a plan that solves the planning problem with security constraints and one solution is given by our previous algorithm, $\mathbf{FAST\_GREEDY}(y^0)$. Otherwise there are no solutions satisfying the BLP policy. We now define our algorithm that takes as inputs the initial state x^0, the initial security labels ℓ^0, the goal g and the security label ℓ_{USER}:

$\mathbf{FAST\_GREEDY\_SEC}(x^0, \ell^0, g, \ell_{USER})$:
1 $y^0 := \mathbf{Reduction}(x^0, \ell^0, \ell_{USER})$;
2 **if** $\mathbf{FG}(y^0) \geq g$ **then** $\mathbf{FAST\_GREEDY}(y^0)$;
else $\mathbf{Return}(\text{“}no\ plan''\text{)}$;

The **Reduction** is the data type selection procedure explained above. It is clear that this reduction from x^0 to y^0 is linear in n. Using inequality (2) and Proposition 1, we can reach the following conclusion.

Proposition 2 *In $\mathcal{O}(m)$ operations the* **FAST_GREEDY_SEC** *algorithm can find a solution to the workflow planning problem with the Bell-LaPadula access constraints, or prove that no solutions exist.*

This result implies that the problem with Bell-LaPadula constraints is not more difficult than the original workflow planning problem without security constraints. However, the extended workflow planning model with multiple labels for each data type that we developed earlier will come into play in the much more complex case where the use of downgraders is allowed.

3.4 Secure Workflows with Downgraders

It has been recognized that the *-property can be overly restrictive in practice. Consequently, a class of trusted processes has been included in the model [Bel74]. These processes are trusted not to violate security policy even though they may violate the *-property. More precisely, a trusted process can have simultaneous read access to the objects of classification ℓ_1 and write access to the objects of classification ℓ_2, even if the label ℓ_2 is not dominating the label ℓ_1 in the lattice.

We define new operators to model the trusted processes that we call downgraders. Let ω^i be a map from $\mathbb{L}$ to $\mathbb{L}^n$ defined by: for $\delta \in \mathbb{L}$, we have $\omega^i(\delta) = (\omega^i(\delta)_1, \ldots, \omega^i(\delta)_n)$ with $\omega^i(\delta)_j = \top$ for $j \neq i$ and $\omega^i(\delta)_i = \delta$. Downgrader $d^i \in D$ acts only on the type $i \in \{1, \ldots, n\}$ and is defined as follows:

$$d^i(x, \ell) := \begin{cases} (x, \ell \wedge \omega^i(\delta_i)) & \text{if } x_i = 1 \\ (x, \ell) & \text{otherwise,} \end{cases} \tag{7}$$

where $\delta_i \in \mathbb{L}$ is a constant label associated with the downgrader d^i. By convention, if there is no downgrader that can act on the data of type i, we define $\delta_i := \top$. We consider the vector of downgrader labels $\delta = (\delta_1, \ldots, \delta_n) \in \mathbb{L}^n$ to be a part of the extended planning problem formulation.

Note that in this model there can exist at most one downgrader per type. However this restriction does not limit the generality of the model: if there are defined several downgraders for the same data type, we can consider the composition of all downgraders for this type as a single downgrader for the purpose of our model.

Proposition 3 *The workflow planning problem under the constraints of the Bell-LaPadula policy with the inclusion of trusted processes (downgraders) is NP-complete.*

The proof of Proposition 3 by reduction to SAT can be found in Appendix B.

4 Conditions For The Existence of Efficient Solutions

Although in this general formulation the secure workflow planning problem is provably hard, we have discovered several special cases in which the problem can be solved efficiently. We dedicate this section to the description of the sufficient conditions for the existence of polynomial time planning algorithms.

4.1 Total Order Condition

We assume in this section that the set of security labels is totally ordered.

For $x \in \{0, 1\}^n$, we define

$$\mathcal{S}(x) := \left\{ \ell \in \mathbb{L} \;\middle|\; \text{such that } (x, \ell) \in \mathcal{S} \right\} \subset \mathbb{L},$$

the subset of all possible labels associated to x. Recall that $\mathcal{S} \neq \{0,1\}^n \times \mathbb{L}^n$ since if $(x,\ell) \in \mathcal{S}$ and type i is not available in x, then we have automatically $\ell_i = \top$.

Following (4) and the subsequent discussion, we assume that the action of an operator on the security labels is given now by

$$\forall j \in A(\theta),\ \forall \ell \in \mathcal{S}(x),\ \overline{\theta}(x,\ell)_j := \bigvee_{k,\, p_k(\theta)=1} \ell_k \wedge \ell_j. \tag{8}$$

Note the presence of $\wedge$ in the right-hand part. By this operation, we take into account the total order assumption on the security labels, i.e. for each type we keep only the lowest security label. Note that the total order assumption allows for some simplifications: in particular, a plan is still well described by a sequence of operators as in Section 2.2 and the generalization given in (5) does not apply here.

In this case the planning problem is significantly simplified since we have the following proposition.

Proposition 4 *For any initial state x^0 and goal g such that $\mathcal{P}(x^0,g)$ is not empty, we can find in time $\mathcal{O}(m)$ a plan π^* such that $\pi^* \in \mathcal{P}(x^0,g)$ and $\forall \pi \in \mathcal{P}(x^0,g),\quad \overline{\pi}(x^0,\ell) \succ \overline{\pi^*}(x^0,\ell)$.*

Proposition 4 implies that it is possible to find a plan that will achieve our goal g while producing data types with the lowest possible security label. Hence once we have $\overline{\pi^*}(x^0,\ell^0) \prec \ell_{USER}$, we have a solution to the planning problem that satisfies the Bell-LaPadula policy with downgraders. On the other hand if $\overline{\pi^*}(x^0,\ell^0) \not\prec \ell_{USER}$, then we know that there is no plan that can solve the planning problem while satisfying the Bell-LaPadula policy. Hence we proved the following proposition:

Proposition 5 *Under the assumption that the lattice of security labels is totally ordered, a workflow that satisfies Bell-LaPadula access policy with downgraders can be composed in linear time $\mathcal{O}(m)$.*

In Appendix C we show that the following plan satisfies the conditions of the Proposition:

$$\pi_{S-OPTI} \;=\; \mathbf{FAST\_GREEDY}(x^0) \;\odot\; D \;\odot\; \mathbf{FAST\_GREEDY}(x^0), \tag{9}$$

Recall that D is the set of downgraders and $\odot$ is the concatenation sign.

We would like to make a few comments on this solution. It is sufficient to apply each downgrader only once, if the position of the downgrader is chosen appropriately. Indeed there are three steps in the algorithm: 1) produce all the possible types (without taking into account the labels); 2) downgrade all the possible labels; 3) produce the optimal labels using regular (non-downgrader) operators.

Another important implication of this proposition is that given the initial types x^0 and the goal g such that $\mathcal{P}(x^0,g)$ is not empty, there exists a plan which achieves the goal g and which will give the minimal possible label of the final state, and this plan does NOT depend on the initial labels ℓ^0. We call this property secure-optimality or s-optimality for short to distinguish it from the standard optimality of plans for STRIPS problems. Indeed the s-optimality of this plan is ensured even if the action of the downgraders is not exactly known a priori. At the planning stage, we do not need to know the values of the δ_i's. Once the plan is deployed, the final value of the labels will depend on these quantities but we know in advance that this plan is s-optimal.

In the case (8), we can compute the label of each type given by any optimal plan. More precisely, as noted at the beginning of this section, we have at most one downgrader d^i for data type i. Then, we have

Proposition 6 *Let π^* be defined as in Proposition 4 and $\ell := \overline{\pi}^*(x^0,\ell^0)$. We denote $\tilde{\ell} := \ell \wedge \delta$. Then,*

$$\ell_i = \bigwedge_{\theta,\, p(\theta) \leq \mathbf{FG}(x^0)} \left(\bigvee_{k,\, p_k(\theta)=1} \tilde{\ell}^0_k \right) \wedge \tilde{\ell}^0_i.$$

Proof.

It follows directly from Proposition 4 that for any plan π^* defined as in Proposition 4, we have $\overline{\pi}^*(x^0, \ell^0) = \overline{\pi}_{S-OPTI}(x^0, \ell^0)$. Hence the claim follows from the description of π_{S-OPTI} given in (9) and the use of (8) and the following distributivity property:

$$\forall a, b, c, \quad a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c). \tag{10}$$

□

4.2 Total Order of Downgrader Labels

Above we proved that if all labels given in the formulation of the planning problem, as well as those obtained after downgrading and combining input labels, are in total order, then a feasible plan can be found in linear time. The simplest special case of this condition is the case where the entire lattice $\mathbb{L}$ of security labels is totally ordered. In this subsection we show that having only a totally ordered set of the downgrader labels $\Delta = \{\delta_i,\ i \in [1, n]\}$ is sufficient to guarantee the existence of an efficient planning algorithm.

We assume now that the set of downgrader labels is totally ordered. In this case, we can number the types in such a way that:

$$\delta_1 \succ \delta_2 \succ \ldots \succ \delta_n \tag{11}$$

In the most general scenario condition (11) may not hold, since downgrader labels are not necessarily comparable. However this assumption is less strict than that of totally ordered lattice, and it yields an efficient planning algorithm. We prove the following proposition Appendix D.1.

Proposition 7 *If the set of downgrader labels is totally ordered, a workflow that satisfies Bell-LaPadula access policy with downgraders can be composed in linear time $\mathcal{O}(m)$.*

Similarly to what is shown in Section 4.1, in the case of ordered downgraders the solution can also be computed independently of input label values, provided that operator input selection in the implementation of (8) can be performed after the plan is created. However, unlike in Section 4.1 the plan will depend on the user label ℓ_{USER}. The security-optimal plan can be constructed by applying the algorithm described in the proof above to the planning problem with all initial labels equal to $\top$.

4.3 Limited Number of Alternative Labels

In many practical cases, the types are not symmetric. For most of the types there exists only one operator producing the type. If this condition holds for every type in the formulation, the plan described above in (9) constitutes the optimal solution.

What happens if several operators produce the same type? Suppose that we run an algorithm that performs an exhaustive search. At some point, it will reach a type that can be produced by at least 2 different operators, and it will have to explore the different possibilities corresponding to these operators. It reached a branching point and the hardness of the problem comes from the number of these branching points. In fact, it can be shown that if the set of branching points does not grow faster than $\log(m)$, then the planning problem can be solved in polynomial time.

Let us now formalize now the idea explained above. Let $B_j := \{\theta \in O \mid a_j(\theta) = 1\}$ be the set of operators that produce the type j. We assume that for all j, we have $|B_j| \leq K$ for some fixed constant K. Let $B := \{j \mid |B_j| \geq 2\}$ be the set of types that are produced by more than one operator.

Proposition 8 *Assume that $|B| \leq B(m)$, then the search cost for the optimal plan is of order $\mathcal{O}(m^2 K^{B(m)})$; if $B(m) = \mathcal{O}(\log(m))$ then this cost is polynomial in m.*

4.4 Downgrader Label Set of Bounded Size

To start, we state the following proposition, the proof of which is given in Appendix D.2.

Proposition 9 *A workflow satisfying Bell-LaPadula access policy with downgraders can be composed in* $\mathcal{O}\left(|\mathbb{L}|^2 m^2\right)$ *operations.*

In practice the size of the label set $\mathbb{L}$ is exponential in the number of security categories $\mathcal{C}$ and is linear in the number of levels $\mathcal{H}$. When written in these terms, the complexity of the algorithm described in Proposition 9 is $\mathcal{O}(2^{2|\mathcal{C}|}|\mathcal{H}|^2 m^2)$. Even if the number of categories remains constant, in practical applications of MLS the set of categories is typically large.

To avoid the dependence on the size of category set, the complexity bound can be expressed in terms of the number of different downgrader classes. Within each class all downgrader share the same value of the downgrader label, and the total number of downgrader classes is equal to the number of distinct downgrader labels. If there are at least two downgraders in the set of downgraders D that have equal downgrader labels, the number of classes will be less than the number of downgraders. The proof of the following proposition is included in the Appendix D.3.

Proposition 10 *The number of operations required to compose a workflow satisfying Bell-LaPadula access policy with downgraders does not exceed* $\mathcal{O}(2^{2^{b+1}}|\mathcal{H}|^2 m^2)$*, where* b *is the cardinality of the set of distinct downgrader labels:*

$$b := |\Delta| = |\{\delta_i, i \in \{1, \ldots, n\}\}|.$$

We note that in practice the approach described in Proposition 10 can be implemented more efficiently, storing bits only for the subsets that are selected during transformation of at least one of ℓ_i^0. The resulting size of the bit vectors can be significantly less than 2^b, which is a highly conservative bound.

Proposition 10 implies that a polynomial solution exists if the number of distinct downgrader labels is limited by a constant. This assumption will often hold in practice, since the number of trusted components in the system tends to be much smaller than the total number of the components due to the high cost of implementing trusted components. The comparatively high implementation costs of downgraders are due to very strict procedural requirements imposed on the development and certification process for trusted components.

5 Workflow Planning with Chinese Wall Constraints

In Section 3 we stated that the workflow planning model we developed is not limited to the Bell-LaPadula MLS policy, and can be applied with any lattice-based security model. The results presented in this section can be seen as an example of such application. We develop a model that can simultaneously represent both the security constraints of the Bell-LaPadula model, and the constraints of so-called Chinese Wall (CW) policy. This model can be also used for workflow planning under only the CW constraints in the applications where the Bell-LaPadula access controls are not enforced. We show that the complexity results derived in previous sections still apply for the extended model under the assumption that the total number of conflict-of-interest classes of the CW policy is bounded by a constant.

5.1 The Chinese Wall Security Policy

The Chinese Wall security policy is a well known information control policy used in the commercial applications and described by Brewer and Nash in [BN89]. It is used to specify control over information when conflicts of interests arise. Information in the computer system is grouped into single objects of the system. These objects are then grouped into datasets, where every object belongs to a single dataset, representing all the information about a single company. The datasets are then classified into conflict of

interest classes. For example, a conflict-of-interest set may contain all of the datasets of oil-companies that employ a certain consultancy firm. Employees of the firm can work with many clients, but cannot work with more than one oil-company without exhibiting a conflict-of-interest.

A user is in violation of the Chinese Wall security policy if she holds information that conflicts with any other information that she already holds. Under this policy, a user who knows nothing is permitted access to any dataset. Once she has obtained information related to a particular dataset, A, a Chinese Wall is built around all datasets that conflict with A. She can still access other information in A and in any other dataset B which is not in conflict with A. After accessing dataset B, the Chinese Wall will be modified to include all datasets in conflict with B.

This policy differs greatly from traditional computer security policies as described in Section 3. However Sandhu has presented in his works [San92], [San93] a Lattice-Based access control model for the Chinese Wall policy which can be represented within the Bell-LaPadula framework. Following this direction, we show now that our framework can address the Chinese Wall policy.

We first recall the framework of [San92]. We begin by distinguishing public information from company information. There are no mandatory controls on reading public information. Reading company information on the other hand is subjected to mandatory controls. Company information is categorized into mutually disjoint conflict of interest classes as shown in Figure 4. Each company belongs to exactly

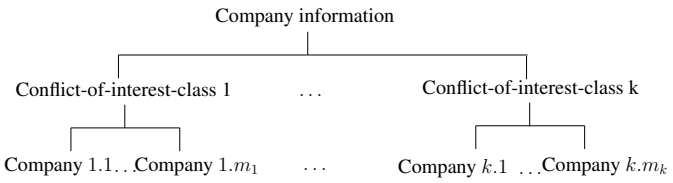

Figure 4: Company information in the Chinese Wall policy.

one conflict of interest (COI) class. The Chinese Wall policy requires that a consultant should not be able to read information for more than one company in any given COI class. The policy for writing public or company information is derived from its potential consequences resulting in indirect read access contrary to mandatory read controls. The policy for writing is essentially the same as the Bell-LaPadula (*-)property with the lattice of labels described below.

Say there are k COI classes: $COI_1, \ldots COI_k$ each with m_j companies. So that $COI_j = \{1, 2, \ldots, m_j\}$, for $j = 1, \ldots, k$. We define a security label as an $n-$ element vector $[i_1, i_2, \ldots, i_k]$ where each $i_j \in COI_j$ or $i_j = \perp$ for $j = 1, \ldots k$. The symbol $\perp$ is read as null. A data labeled $[i_1, i_2, \ldots, i_k]$ is interpreted as (possibly) containing information from company i_1 of COI_1, company i_2 of COI_2 and so on. When an element of the label is $\perp$ rather than an integer, the data cannot contain any information from any company in the corresponding COI class.

The dominance relation among labels is defined as follows: $s_1 \prec s_2$ provided s_1 and s_2 agree wherever $s_1 \neq \perp$. To be precise, let $s(i_j)$ denote the i_j-th element of label s. Then $s_1 \prec s_2$ if $s_1(i_j) = s_2(i_j)$ or $s_1(i_j) = \perp$, for all $j = 1, \ldots k$.

Public data is labeled $[\perp, \ldots, \perp]$. Moreover we introduce a special label $\top$ (called Syshigh in [San92]) that dominates all other labels. This label is contrary to the Chinese Wall policy, which we will explain later on. We define now the upper bound in our lattice. First we say that two labels s_1 and s_2 are compatible if wherever they disagree at least one of them is $\perp$, that is, $s_1(i_j) = s_2(i_j)$ or $s_1(i_j) = \perp$ or $s_2(i_j) = \perp$ for all $j = 1, \ldots, k$. Incompatible labels cannot be legitimately combined under the Chinese Wall policy. This is expressed by the requirement that if s_1 is incompatible with s_2, then $s_1 \vee s_2 = \top$. For compatible labels, we have

$$(s_1 \vee s_2)(i_k) = \textbf{if } s_1(i_k) \neq \perp \textbf{ then } s_1(i_k) \textbf{ else } s_2(i_k).$$

For example $[1, \perp, 2] \vee [1, 2, \perp] = [1, 2, 2]$. Finally, the upper bound of any label with $\top$ is $\top$.

Given this lattice structure, Sandhu introduces the concepts of *users*, *principals* and *subjects* in order to describe how the Chinese Wall policy can be enforced. Each human being known to the system is recognized as an unique user. Every time a user logs into the system it is as a particular principal. For example, there is an unique user John, cleared to top-secret, independent of the level at which John logs in. John can log in at every level dominated by top-secret. At each of these levels there is a separate principal associated with John. A subject is a process in the system, i.e. a program in execution. Each subject is associated with a single principal on behalf of whom the subject executes. In general a principal may have many subjects associated with it concurrently running in the system. We refer to [San92] for more details on these concepts.

To describe how the Chinese Wall policy is enforced, we follow the example presented in [San92] in the context of the specific lattice of Figure 5, which contains two COI classes with two companies in each class.

Objects labeled $\top$ violate the Chinese Wall policy by combining information from more than one company in the same COI class. These objects are inaccessible in the system, since no user will be cleared to $\top$. Now let us consider the labels on users, principals and subjects. We treat the label of a user as a high-water mark which can float up in the lattice, but not down. A newly enrolled user in the system is assigned the label $[\bot, \bot]$. As the user accesses various company information, the user's label floats up in the lattice. For example, by accessing information about company 1 in COI class 1 the user's label is modified to $[1, \bot]$. reading information about company 2 in COI class 2 further modifies the user's label to $[1, 2]$. This floating up of a user's label is allowed, as long as the label does not float up to $\top$. Operations which would force the user's label to $\top$ are thereby prohibited.

With each user we associate a set of principals, one at each label dominated by the user's label. Thus if Jane as a user has the label $[1, 1]$, she has the following principals associated with her: Jane.$[1, 1]$, Jane.$[1, \bot]$, Jane.$[\bot, 1]$ and Jane.$[\bot, \bot]$. Each of these corresponds to the label with which she wishes to log in on a given session. Each principal has a fixed label which does not change. Every subject created by that principal inherits that label. All read and write operations in the system are carried out by subjects. These subjects are constrained by the (ss-) and (*-)properties of the Bell-LaPadula model. For example, suppose that Jane logs in as the principal $[1, \bot]$. All subjects created during that session will inherit the label $[1, \bot]$. This will allow theses subjects to read public data labeled $[\bot, \bot]$, to read and write company objects labeled $[1, \bot]$ and write objects with labels $[1, 1]$, $[1, 2]$ and $\top$.

5.2 Planning Secure Workflows for The Chinese Wall Policy

In our discussion of the planning approach we will use the example lattice shown on Figure 5. However the solution we develop is general and will be scalable in the sense that all the algorithms will run in a linear time in m as soon as the number of COI classes and the number of companies in each of these classes is bounded by a constant C that is independent of m.

The framework of Section 2 is still valid here, but we have now to modify the one of Section 3. First note that the security label of a user ℓ_{USER} is not fixed anymore. We will assume that the security label of a new user is $[\bot, \bot]$, and we will update it as needed during planning.

We still consider the state space $\mathcal{S}$ defined in (3) where $(\mathbb{L}, \prec)$ is now the Chinese Wall lattice of Figure 5. The action of an operator is still given by (4) that we recall here:

$$\forall j \in A(\theta),\ \forall \ell \in \mathcal{S}(x),\ \overline{\theta}(x, \ell)_j := \bigvee_{k,\, p_k(\theta)=1} \ell_k.$$

Note in particular that if two labels of the input are not compatible, then the output label is $\top$.

For any variable $(x, \ell) \in \mathcal{S}$ and any security label s, we define $x[s]$ as follows

$$x_i[s] = \textbf{if } \ell_i \prec s \textbf{ then } x_i \textbf{ else } 0.$$

For example $x[\bot, \bot]$ corresponds to the data that are present in the state x and is public.

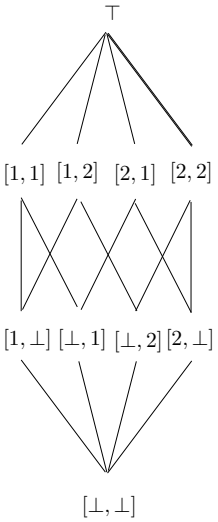

Figure 5: Example of a Chinese Wall lattice (Hasse diagram).

Suppose now that a new user logs into the system and asks for a plan in $\mathcal{P}(x^0, g)$. We assume that $\mathcal{P}(x^0, g) \neq \emptyset$, the question addressed here is whether there exists a secure plan or not. We define the following variables associated to the labels of Figure 5,

$$x^0[\bot, \bot], x^0[1, \bot], x^0[\bot, 1], x^0[\bot, 2] \ldots$$

Note that we have for example $x^0[1, 2] = x^0[1, \bot] \vee x^0[\bot, 2]$.

If $\mathbf{FG}(x^0[\bot, \bot]) \geq g$, then $\mathbf{FAST\_GREEDY}(x^0[\bot, \bot])$ is a solution that uses only public information. Hence this solution satisfies clearly the Chinese Wall policy and it should be given to the user. Suppose now that $\mathbf{FG}(x^0[\bot, \bot]) \not\geq g$, this means that the goal cannot be attained with the public information of this user.

If $\mathbf{FG}(x^0[1, \bot]) \geq g$, then $\mathbf{FAST\_GREEDY}(x^0[1, \bot])$ is a solution that is valid for the user since her label is $[\bot, \bot]$. But her label has to be updated to $\ell_{USER} = [1, \bot]$. Note in particular that the data used by this plan will have the label $[\bot, \bot]$ or $[1, \bot]$, hence the Chinese Wall policy is satisfied once we update the user's label. The situation is similar for $x^0[2, \bot]$ or $x^0[\bot, i]$ with $i = 1, 2$.

If none of the $\mathbf{FAST\_GREEDY}(x^0[i, j])$ where either i or j is set to $\bot$ gives a solution, then we have to go one step further and test if $\mathbf{FG}(x^0[i, j]) \geq g$ where $i, j = 1, 2$. Suppose that for example $\mathbf{FG}(x^0[1, 2]) \geq g$, then we can give the corresponding plan to the user and update her security label to $\ell_{USER} = [1, 2]$. In this case, the data used by the plan will have the labels $[\bot, \bot]$, $[1, \bot]$, $[\bot, 2]$, $[1, 2]$. Hence the Chinese Wall policy is still enforced.

If none of the $\mathbf{FAST\_GREEDY}(x^0[i, j])$ gives a solution, this implies that any plan in $\mathcal{P}(x^0, g)$ uses data from two companies within the same COI class. Hence we conclude that there exists no secure plan.

Now assume that the user already has been assigned a security label $\ell_{USER} \neq [\bot, \bot]$. The same algorithm as above will work but we have now to consider the $x^0[s]$ for s compatible with ℓ_{USER} and $s \neq \top$. For example, if $\ell_{USER} = [1, \bot]$, then we have to consider:

$$x^0[\bot, \bot],\ x^0[1, \bot],\ x^0[1, 1],\ x^0[1, 2].$$

We can now write our algorithm in a compact way:

CHINESE_WALL(x, g, ℓ_{USER}) :
1 **if** $\mathbf{FG}(x) \not\geq g$ **then Return**("*no plan*");
2 **for** s compatible with ℓ_{USER} **do**
3 **if** $\mathbf{FG}(x[s]) \geq g$ **then**
Return(**FAST_GREEDY**$(x[s]), \ell_{USER} \vee s$);
od
4 **Return**("*no secure plan*");

Since the number of security labels is bounded, thanks to Proposition 1 we have:

Proposition 11 *A workflow that satisfies the Chinese Wall policy can be composed in linear time* $\mathcal{O}(m)$.

Note that the algorithm **CHINESE_WALL** will give a plan that satisfies the Chinese Wall policy and the updated security label of the user. If this security label is not updated to the value returned by this algorithm, the Chinese Wall policy will not be enforced at the next query of the user. The exact manner in which a user's clearance is allowed to float up is not specified in our model, since there are numerous alternatives. For example, assume that $\mathbf{FG}(x[s]) \geq g$ and $\mathbf{FG}(x[s']) \geq g$ for two distinct labels s, s' compatible with ℓ_{USER}. Then if the algorithm return the plan corresponding to s (resp. s') then the user's clearance has to be updated to $\ell_{USER} \vee s$ (resp. to $\ell_{USER} \vee s'$). It seems clear that if $s \prec s'$, then the solution corresponding to s is better in the sense that the user is exposed to less conflicting information. In particular note that if $\mathbf{FG}(x[1, \bot]) \geq g$ then we have automatically $\mathbf{FG}(x[1, 1]) \geq g$ but we will prefer the solution associated to $[1, \bot]$ since it leaves the freedom for the user to access any information from COI 2 in the future. Hence the loop **for** of step 2 should be done such that the successive value of s are non-decreasing.

5.3 Extension

In this section, we describe an extension of the Chinese Wall policy for the case when each company has a security policy based on the Bell-LaPadula model. We call this security policy the CW-BLP policy.

We assume now that there is a category associated to each user. For example any user of the system is known as a systems analyst (category A) or a security analyst (category B).

The structure of the tree depicted in Figure 4 gives the COI between each company. But now, in addition to it, each company applies on its own data, its own security policy based on a totally ordered set of hierarchical levels. For simplicity we will assume that all the companies have the same set of labels (but the general case can be handled in the same way):

$$\mathcal{H} = \{Public \leq_{\mathcal{H}} Classified \leq_{\mathcal{H}} Secret \leq_{\mathcal{H}} Top - Secret\}.$$

For each category, the company allows the corresponding user to be cleared at a certain level. For example the company i of COI_j will clear any user of category A to $Top - Secret$ and any user of category B to $Secret$. We will denote it by $A[j.i] = Top - Secret$ and $B[j.i] = Secret$.

Now each company applies the Bell-LaPadula policy as described in Section 3:

- a user is allowed to read data only if the label of the user dominates the label of the data;
- a user is allowed to write data only if the label of the data dominates the label of the user.

Following what we did in Section 3, data κ has now a security label of the following form:

$$\ell_\kappa = (\ell_\kappa(\mathbb{L}), \Lambda_\kappa) := ([i_1, i_2, \ldots, i_k], [\lambda_1, \lambda_2, \ldots, \lambda_k]),$$

where each $i_j \in COI_j$ or $i_j = \perp$ and each $\lambda_j \in \mathcal{H}$. We still denote the Chinese Wall lattice by $(\mathbb{L}, \prec)$ and Λ_κ belongs to $\mathcal{H}^k$. We assume moreover that $i_j = \perp$ implies $\lambda_j = Public$. The interpretation of $[i_1, i_2, \ldots, i_k]$ is the same as in Section 5.1. For example the raw input data of company i in COI_2 will have a label of the form

$$([\perp, i, \perp, \ldots, \perp], [Pub., Secret, Pub., \ldots, Pub.]).$$

In order to model the Bell-LaPadula security inside each company, we use for each operator the same rules as described in Section 3. First note that the security label of a user is completely determined by $\ell_{USER}(\mathbb{L}) \in \mathbb{L}$ (with the same interpretation as in previous section) and her category. We assume that the action of an operator is still given by Equation (4). Given a state variable (x, ℓ) and a label $s \in \mathbb{L}$, we define now:

$$\begin{aligned}(x_\kappa[s], \ell_\kappa[s]) \quad = \quad & \textbf{if } \ell_\kappa(\mathbb{L}) \prec s \textbf{ then } (x_\kappa, \ell_\kappa) \\ & \textbf{else } (0, (\top, [Pub.]).\end{aligned}$$

To enforce the Chinese Wall policy, we will ensure as in previous section that each time we search for a plan, the search is restricted to $(x[s], \ell[s])$ for a certain $s \in \mathbb{L}$.

Assume for simplicity that the Chinese Wall lattice is the one shown on Figure 5 and there are two categories A and B. Suppose now that a new user of category A is logging in the system and asking for a plan in $\mathcal{P}(x^0, g) \neq \emptyset$. As in previous case, if $\mathbf{FG}(x^0[\perp, \perp]) \geq g$, then $\mathbf{FAST\_GREEDY}(x^0[\perp, \perp])$ is a solution that uses only public information and this solution should be given to the user. Suppose now that $\mathbf{FG}(x^0[\perp, \perp]) \not\geq g$ and $\mathbf{FG}(x^0[1, \perp]) \geq g$. It gives a solution that satisfies the Chinese Wall policy. However, before presenting this plan to the user, we have to check if the plan uses data that should not be exposed to the user. Indeed we should check if the algorithm $\mathbf{FAST\_GREEDY\_SEC}(x^0[1, \perp], \ell^0[1, \perp], g, [A[1.1], Pub.])$ returns a plan. If it does, we can return this plan to the user and update her security label to $([1, \perp], [A[1.1], Pub.])$. It may happen that we have to go one step further, and in general we have to follow the algorithm given below:

$\mathbf{CW\_BLP}(x, \ell^0, g, \ell_{USER}(\mathbb{L}), C)$:
1 **if** $\mathbf{FG}(x) \not\geq g$ **then** $\mathbf{Return}(\text{“}no\ plan''\text{)}$;
2 **for** s compatible with $\ell_{USER}(\mathbb{L})$ **do**
  $\pi := \mathbf{FAST\_GREEDY\_SEC}(x[s], \ell^0[s], g, C(s))$;
3 **if** π is a plan **then**
    $\mathbf{Return}(\pi, \ell_{USER}(\mathbb{L}) \vee s)$;
  od
4 $\mathbf{Return}(\text{“}no\ secure\ plan''\text{)}$;

The inputs of the algorithm are x the initial sate, g the goal, ℓ_{USER} the current security label of the user and C her category (i.e. A or B). We used the following notation:

$$C([i_1, i_2, \ldots, i_k]) = [C[1.i_1], \ldots C[k.i_k]],$$

with the convention that $C[j.\ \perp] = Public$.

Hence, we have thanks to Proposition 2:

Proposition 12 *A workflow that satisfies the CW-BLP policy can be composed in linear time $\mathcal{O}(m)$.*

Note that the same kind of remarks as in previous section applies here too. Note further that the concept of a downgrader in this framework is not clear in general. However, each company may have downgraders that act only on labels of the following type

$$([\bot, \ldots, i, \ldots, \bot], [Pub., \ldots, Top - Secret, \ldots, Pub.])$$

to give the same data but with the following label

$$([\bot, \ldots, i, \ldots, \bot], [Pub., \ldots, Secret, \ldots, Pub.]).$$

In this case, one can use the results of Section 4.1 to see that Proposition 12 still holds with that kind of downgraders.

6 Conclusion

In this paper we have analyzed the computational complexity of problems arising in workflow planning under security constraints. These results were developed for the automatic building and deployment of applications in System S, a large scale distributed stream processing system. We have shown that in many practical scenarios the planning problem can be solved efficiently. Our analysis framework and a set of basic results can be easily extended for many lattice-based access control policies. In this paper we use the framework to analyze workflow planning problem under the Bell-LaPadula and the Chinese Wall policies. We describe an efficient (linear time) algorithm for planning under these constraints. We also show that if the planning problem becomes much harder (NP-complete) if the use of downgraders is allowed. Nevertheless, under certain assumptions planning can be performed efficiently even with downgraders. We identify several classes of assumptions, and describe corresponding efficient planning algorithms.

The results we present in this paper show that the use of automatic planning techniques within compositional environments with lattice-based access controls constitutes not only an attractive, but also a practically feasible approach.

In this work we have explored only one dimension of context-dependent constraints on workflows. This allowed us to identify and describe the classes of security constraints that correspond to different complexity classes of planning problems. However, in practical systems the security constraints must be used together with other types of context-dependent constraints, such as semantic or resource constraints. All these constraints further increase the complexity. For example, adding simple resource utilization constraints based on an additive resource metric immediately makes the problem NP-hard by reduction to SET COVER. The work in this area currently focuses, and in the near future will continue to focus, on the design of efficient planning algorithms that can support a wide range of composition constraints.

Acknowledgements

We thank our colleagues Pankaj Rohatgi and Pau-Chen Cheng for very helpful discussions at the beginning of this work.

References

[AVMM04] R. Aggarwal, K. Verma, J. A. Miller, and W. Milnor. Constraint driven web service composition in METEOR-S. In *Proceedings of SCC-04*, 2004.

[BDG+03] J. Blythe, E. Deelman, Y. Gil, K. Kesselman, A. Agarwal, G. Mehta, and K. Vahi. The role of planning in grid computing. In *Proceedings of ICAPS-03*, 2003.

[Bel74] D.E. Bell. Secure computer systems: A refinement of the mathematical model. *MTR-2547, Vol. III, MITRE Corp.*, 1974.

[Bib77] K.J. Biba. Integrity considerations for secure computer systems. *MTR-3153, MITRE Corp.*, 1977.

[BKH05] A. Brown, A. Keller, and J. Hellerstein. A model of configuration complexity and its application to a change management system. In *Proceedings IM-05*, 2005.

[BL76] D.E. Bell and L.J. LaPadula. Secure computer system: Unified exposition and Multics interpretation. *MTR-2997, MITRE Corp.*, 1976.

[BN89] D. F. C. Brewer and M. J. Nash. The Chinese Wall security policy. *Proc. IEEE Symposium on research in security and privacy*, pages 206–214, 1989.

[Byl94] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

[CM97] S. A. Cook and D. G. Mitchell. Finding hard instances of the satisfiability problem: a survey. In *Satisfiability problem: theory and applications (Piscataway, NJ, 1996)*, volume 35 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 1–17. Amer. Math. Soc., Providence, RI, 1997.

[Den76] D.E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, May 1976.

[DGAV04] P. Doshi, R. Goodwin, R. Akkiraju, and K. Verma. Dynamic workflow composition using Markov decision processes. In *Proceedings of ICWS-04*, 2004.

[FHN72] R. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(1-3):251–288, 1972.

[GDB+04] Y. Gil, E. Deelman, J. Blythe, C. Kesselman, and H. Tangmurarunkit. Artificial intelligence and grids: Workflow planning and beyond. *IEEE Intelligent Systems*, January 2004.

[KIK03] T. Kichkaylo, A. Ivan, and V. Karamcheti. Constrained component deployment in wide-area networks using AI planning techniques. In *Proceedings of IPDPS-03*, 2003.

[KS03] J. Koehler and B. Srivastava. Web service composition: Current solutions and open problems. In *Proceedings of ICAPS-03, Workshop on Planning for Web Services*, pages 28–35, 2003.

[MPM+04] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara. Bringing semantics to Web Services: The OWL-S approach. In *Proceedings of SWSWPC-04*, 2004.

[PTB05] M. Pistore, P. Traverso, and P. Bertoli. Automated composition of web services by planning in asynchronous domains. In *Proceedings of ICAPS-05*, 2005.

[RH01] J. Rintanen and J. Hoffmann. An overview of recent algorithms for AI planning. *Künstliche Intelligenz*, (2):5–11, May 2001.

[RL05] A. Riabov and Z. Liu. Planning for stream processing systems. In *Proceedings of AAAI-05*, 2005.

[San92] R. Sandhu. A lattice interpretation of the Chinese Wall policy. *Proc. of the 15th NIST-NCSC National Computer Security Conference*, pages 221–235, 1992.

[San93] R. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, 1993.

[WPS+03] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S web services composition using SHOP2. In *Proceedings of ISWC2003*, 2003.

A Proof of Proposition 1

Proposition 1. *Given a set of m operators, for any initial condition x^0 and a goal g, in $\mathcal{O}(m)$ operations an element of $\mathcal{P}(x^0, g)$ can be found if $\mathcal{P}(x^0, g) \neq \emptyset$. If such an element is not found by this procedure, then necessarily $\mathcal{P}(x^0, g) = \emptyset$.*

We assume that we have a table of size n such that the i-th entry points to the set of operators that have i as precondition, namely

$$Pre(i) = \{\theta \in O \mid p_i(\theta) = 1\} \subset O.$$

We first show that the algorithm **FAST_GREEDY** runs in linear time. As we store the preconditions of each operator $P(\theta)$ as a list of C variables at most, Step 7 can be performed in $\mathcal{O}(1)$ time. The set W will hold a worklist and all operations on W can be made in $\mathcal{O}(1)$ time. Each type enters the worklist at most once and each iteration of the while loop removes one type from the worklist. Consequently, the number of iterations of the while loop is at most the number of types and so the total number of executions of Steps 4-5 is at most $\mathcal{O}(n)$. Each operator is examined at most C times in Steps 7-8, hence the total number of executions of Steps 7-8 is at most $\mathcal{O}(m)$ with a careful updating of the sets $Pre(i)$. Adding up all the costs, we see that the total running time of **FAST_GREEDY** is $\mathcal{O}(m + n)$, where m is the number of operators and n the number of types.

Intuitively the state $\mathbf{FG}(x^0)$ describes all the types that can be produced from the initial state x^0. Hence we must have that

$$\forall \pi,\ \pi(x^0) \leq \pi_{FG}(x^0) = \mathbf{FG}(x^0), \tag{12}$$

which would imply the claim since it implies that $\mathbf{FG}(x^0) \geq g \Leftrightarrow \mathcal{P}(x^0, g) \neq \emptyset$ and in this case, $\pi_{FG} \in \mathcal{P}(x^0, g)$.

Indeed, as it turns out, to prove Proposition 1 only the following basic properties of the operators are required:

Lemma 1

1. $\forall \theta \in O,\ \theta \cdot \theta = \theta$ *(idempotency);*
2. $\forall x,\ x \leq \theta(x)$ *(non-decreasingness);*
3. $\forall \theta^1, \theta^2$, *such that* $x \geq \max(p(\theta^1), p(\theta^2))$, $\theta^1 \cdot \theta^2(x) = \theta^2 \cdot \theta^1(x)$ *(local commutativity).*

Note that we have not in general commutativity, and that the last property depends on the state of the system x.

We now prove (12). We denote by $\pi_1(= \emptyset), \pi_2, \ldots$ the successive values of π before the first, second, ... while loops and $u_i = \pi_i(x^0)$. Note that we have $u_1 \leq u_2 \leq \cdots \leq FG(x^0)$.

Idempotency and local commutativity imply that for all operators θ,

$$\begin{aligned} \theta(\mathbf{FG}(x^0)) &= \theta \cdot \pi_{FG}(x^0) = \pi_{FG}(x^0) \\ &= \mathbf{FG}(x^0), \end{aligned}$$

this is clear on the event $\neg\{\mathbf{FG}(x^0) \geq p(\theta)\}$, where $\neg$ stands for negation. If $\mathbf{FG}(x^0) \geq p(\theta)$, then let i be the minimal j such that $u_j \geq p(\theta)$. We have

$$\begin{aligned}\theta \cdot \pi_{FG}(x^0) &= \theta \cdot \theta^{\alpha_1} \cdot \cdots \cdot \theta \cdot \theta^{\beta_1} \cdot \cdots \pi_{i-1}(x^0) \\ &= \theta^{\alpha_1} \cdot \cdots \cdot \theta \cdot \theta \cdot \theta^{\beta_1} \cdot \cdots \pi_{i-1}(x^0) \\ &= \theta^{\alpha_1} \cdot \cdots \cdot \theta \cdot \theta^{\beta_1} \cdot \cdots \pi_{i-1}(x^0) \\ &= \pi_{FG}(x^0).\end{aligned}$$

Now we have

$$\begin{aligned}x^0 &\leq \pi_{FG}(x^0) \\ \pi(x^0) &\leq \pi \cdot \pi_{FG}(x^0) = \pi_{FG}(x^0),\end{aligned}$$

which is (12).

B Proof of Proposition 3

Proposition 3. *The workflow planning problem under the constraints of the Bell-LaPadula policy with the inclusion of trusted processes (downgraders) is NP-complete.*

Proof. We will show that given an instance of NP-complete SATISFIABILITY problem (SAT, see [CM97] for details) we can construct an instance of workflow planning problem with downgraders, the optimal solution to which can be used to construct a solution for SAT. All transformations involve polynomially many steps and size of the planning problem is polynomial in the size of the SAT instance.

Given a set of Boolean variables $\{x_1, x_2, \ldots, x_n\}$, satisfiability problem is to find values for these variables that makes true a given Boolean formula

$$F(x) := \bigwedge_{i=1}^{q} \left\{ \left(\bigvee_{j=1}^{m_i} x_{I(i,j)} \right) \vee \left(\bigvee_{j=1}^{p_i} \overline{x}_{J(i,j)} \right) \right\},$$

written in conjunctive normal form. We will denote

$$I(i) := \bigcup_{j=1}^{m_i} I(i,j), \quad J(i) = \bigcup_{j=1}^{p_i} J(i,j).$$

We define the state space $\mathcal{S}$ as in (3) such that $\mathcal{S} \subset \{0,1\}^{3n+1} \times \mathbb{L}^{3n+1}$, with $\mathbb{L} := \{0,1\}^q \cup \{\top\}$. For each type $2i$ and $3i$ with $i \in \{1, n\}$, we define a downgrader as described in equation (7) where the corresponding δ's are defined for all $1 \leq j \leq q$ as

$$\delta_j^{2i} := 1 - \mathbb{1}_{\{j \in I(i)\}}, \quad \delta_j^{3i} := 1 - \mathbb{1}_{\{j \in J(i)\}}.$$

Above, $\mathbb{1}_{\{\}}()$ is an indicator function that is equal to 1 in the condition in the subscript is true, and 0 otherwise. Recall that for $i \in \{1, 3n+1\}$, $e^i \in \{0,1\}^{3n+1}$ is defined by $e^i_j = 0$ for $j \neq i$ and $e^i_i = 1$. We now define the operators that from e^i produce e^{2i} and e^{3i} for $i \in [1, n]$, namely operators t^i and f^i respectively are defined as follows:

$$p(t^i) = e^i,\ a(t^i) = e^{2i}, \quad p(f^i) = e^i,\ a(f^i) = e^{3i}.$$

We define now for each $i \in [1, n]$, two other operators ot^i and of^i as follows:

$$p(ot^i) = e^{2i},\ a(ot^i) = e^{i+1}, \quad p(of^i) = e^{3i},\ a(of^i) = e^{i+1}.$$

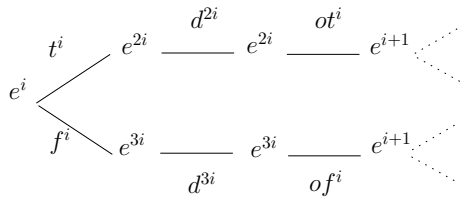

Figure 6: Reduction to SAT.

We consider the following planning problem with the initial condition e^1 and with the goal e^{n+1}. We assume that the initial security label is $\ell = (1, \ldots, 1)$ and that the security level of the user is $\ell_{USER} = (0, \ldots, 0)$.

In total, we defined $4n$ operators and $2n$ downgraders where n is the number of types and the size of x in the SAT instance $F(x)$ (see Figure 6).

It is easy to see that if SAT has a solution, there will exist a plan solving the planning problem with security constraints: for each variable we will either apply t^k or f^k, but not both, which corresponds to setting variable x_k to either "true" or "false". On the other hand, if a plan solves the planning problem, we can construct a solution to SAT. Hence, the workflow planning problem is NP-complete.

□

C Proof of Proposition 4

Proposition 4. *For any initial state x^0 and goal g such that $\mathcal{P}(x^0, g)$ is not empty, we can find in time $\mathcal{O}(m)$ a plan π^* such that $\pi^* \in \mathcal{P}(x^0, g) \quad$ and $\quad \forall \pi \in \mathcal{P}(x^0, g), \quad \overline{\pi}(x^0, \ell) \succ \overline{\pi^*}(x^0, \ell)$.*

We first describe the properties that are satisfied by each operator, and that guarantee the correctness of our algorithm.

Lemma 2 *The action of an operator θ on the labels is such that*

1. *Idempotency: for all operators θ,*

$$\forall \ell \in \mathcal{S}(x),\ \theta \cdot \theta(x, \ell) = \theta(x, \ell). \tag{13}$$

2. *Local commutativity: for any x and θ^1, θ^2 such that $x \geq p(o^i)$, we have*

$$\forall \ell \in \mathcal{S}(x),\ \theta^1 \cdot \theta^2(x, \ell) = \theta^2 \cdot \theta^1(x, \ell); \tag{14}$$

3. *Monotonicity: for all x,*

$$\forall \ell^1, \ell^2 \in \mathcal{S}(x),\ \ell^1 \prec \ell^2 \Rightarrow \overline{\theta}(x, \ell^1) \prec \overline{\theta}(x, \ell^2); \tag{15}$$

4. *Non-increasingness: for all x,*

$$\forall \ell \in \mathcal{S}(x),\ \overline{\theta}(x, \ell) \prec \ell; \tag{16}$$

Proof.

The only non-trivial assumption to check is the commutativity condition (14). But in this case, thanks to distributivity (10), we have

$$\begin{aligned} \overline{\theta^1 \cdot \theta^2}(\ell)_j &= \left(\bigvee_{k,\, p(\theta^1)_k = 1} \ell_k \right) \wedge \left(\bigvee_{i,\, p(\theta^2)_i = 1} \ell_i \right) \wedge \ell_j \\ &= \overline{\theta^2 \cdot \theta^1}(\ell)_j, \end{aligned}$$

from which the condition (14) follows. □

It is clear that the set of downgraders satisfies the properties (15), (16), (14) and (13). But in general we have

$$\theta \cdot d(x, \ell) \neq d \cdot \theta(x, \ell).$$

Since downgraders do not affect the type, we have

$$\underline{\theta \cdot d}(x, \ell) \quad = \quad \underline{\theta}(x, \ell).$$

Moreover thanks to (15) we have

$$\overline{\theta \cdot d}(x, \ell) \quad \prec \quad \overline{\theta}(x, \ell).$$

Note that if x is such that $\underline{\theta}(x) = x$, we have for all $d \in D$,

$$d \cdot \theta \cdot d = \theta \cdot d. \tag{17}$$

We now show that π_{S-OPTI} as defined in (9) satisfies the condition of Proposition 4. Thanks to Proposition 1, we have

$$\pi_{S-OPTI}(x^0) \geq g \Leftrightarrow \mathcal{P}(x^0, g) \neq \emptyset.$$

For simplicity, we denote $u = \pi_{S-OPTI}(x^0)$.

Thanks to properties (14) and (13), we see that for all operators such that $u \geq p(\theta)$, we have,

$$\theta \cdot \pi_{S-OPTI}(x^0, \ell) = \pi_{S-OPTI}(x^0, \ell).$$

Thanks to property (17), we have for $d \in D$,

$$d \cdot \pi_{S-OPTI}(x^0, \ell) = \pi_{S-OPTI}(x^0, \ell).$$

Hence for any plan $\pi \in \mathcal{P}(x^0, g)$, we have

$$\pi \cdot \pi_{S-OPTI}(x^0, \ell) = \pi_{S-OPTI}(x^0, \ell). \tag{18}$$

Moreover thanks to (16), we have

$$\forall \ell \in \mathcal{S}(u), \quad \overline{\pi_{S-OPTI}}(x^0, \ell) \prec \ell,$$

hence thanks to (15), we have

$$\overline{\pi}\left(x^0, \overline{\pi_{S-OPTI}}(x^0, \ell)\right) \prec \overline{\pi}(x^0, \ell).$$

Now it is easy to see that

$$\begin{aligned} \overline{\pi}\left(x^0, \overline{\pi_{S-OPTI}}(x^0, \ell)\right) &= \overline{\pi}\left(\pi_{S-OPTI}(x^0, \ell)\right) \\ &= \overline{\pi_{S-OPTI}}(x^0, \ell), \end{aligned}$$

thanks to (18). Hence we proved that for any $\pi \in \mathcal{P}(x^0, g)$, we have

$$\overline{\pi}(x^0, \ell) \succ \overline{\pi_{S-OPTI}}(x^0, \ell),$$

and the proposition follows.

D Proofs and Algorithms for Special Cases

D.1 Proof of Proposition 7

Proposition 7. *If the set of downgrader labels is totally ordered, a workflow that satisfies Bell-LaPadula access policy with downgraders can be composed in linear time $\mathcal{O}(m)$.*

Proof.

Combining (11) and (7) it is easy to see that if in some state (x, ℓ) the labels are such that $\ell_i = \ell_j$ for some $i \leq j$, then after applying the downgraders d^i and d^j in the new state (x', ℓ') we will have $\ell'_i \succ \ell'_j$. Therefore if $\ell'_i \prec \ell_{USER}$, then $\ell'_j \prec \ell_{USER}$ as well. This property allows us to define a new order relation $\trianglelefteq$ on the labels, under which the entire set of labels will become totally ordered. To compare arbitrary labels ℓ'_i and ℓ''_i obtained for the same type, we will compute a scalar *downgrader index* $\mathcal{I}(\ell)$ for each label, and compare the indices. Formally, the precedence $\ell'_i \trianglelefteq \ell''_i$ holds if and only if $\mathcal{I}(\ell') \leq \mathcal{I}(\ell'')$, where $\mathcal{I}(\ell')$ is the smallest index of the downgrader that makes the label accessible to the user:

$$\mathcal{I}(\ell) := \begin{cases} 0, & \ell \prec \ell_{USER}; \\ (k+1), & \ell \wedge \delta_k \not\prec \ell_{USER}; \\ \min_i \{ i \colon \ell \wedge \delta_i \prec \ell_{USER} \}, & \text{otherwise.} \end{cases} \tag{19}$$

In this partial order it is possible to define the label transformation formula for operators similarly to (8). If alternative labels for the same type are computed, the smallest label now must be chosen according to the new order relation $\trianglelefteq$. Since the union operation $\vee$ is well defined for the elements of partially ordered lattice $\mathbb{L}$, we only need to define the minimum operation $\wedge$. In the following formula $\tilde{\wedge}$ computes the lower bound of two labels in $\trianglelefteq$ order.

$$\forall j \in A(\theta),\ \forall \ell \in \mathcal{S}(x),\ \overline{\theta}(x, \ell)_j := \left(\bigvee_{k,\, p_k(\theta)=1} \ell_k \right) \tilde{\wedge} \ell_j. \tag{20}$$

With this revision of label transformation, Lemma 2 still holds and π_{S-OPTI} algorithm can now be applied, as specified in (9).

The union operation $\vee$, as defined in the lattice $\mathbb{L}$, directly corresponds to the upper bound in $\trianglelefteq$ lattice, and therefore the correctness of this algorithm follows immediately from the correctness of π_{S-OPTI}. As shown in Section 4.1, the algorithm will find a plan π^* that will produce the minimum output label. Due to modification (20), the label that it finds for each type will be the smallest possible with respect to the order $\trianglelefteq$.

According to the definition of $\trianglelefteq$ in (19), the label ℓ in $\trianglelefteq$ is the smallest in the order if and only if $\mathcal{I}(\ell) = 0$, which can hold if and only if goal condition $\ell \prec \ell_{USER}$ is satisfied for ℓ. Therefore, the π_{S-OPTI} algorithm (9) with a modified order relation as specified in (20) will find a plan that satisfies the goal or show that no such plans exist. □

D.2 An Algorithm Polynomial in the Size of the Label Set

In Section 4.3 we showed that restricting the set of branching points to grow at most as a logarithm of the number of operators m results in polynomial complexity of exhaustive search. Here we will describe a dynamic programming algorithm, complexity of which does not depend on the number of branching points defined above, but is at most quadratic in the cardinality of the label lattice $\mathbb{L}$ and in the number of operators m. We will prove the following proposition.

Proposition 9. *A workflow satisfying Bell-LaPadula access policy with downgraders can be composed in $\mathcal{O}\left(|\mathbb{L}|^2 m^2\right)$ operations.*

Let $\mathcal{F}_i$ be the set of alternative labels for type i, $\mathcal{F}_i \subseteq \mathbb{L}$. The algorithm maintains the set $\mathcal{F}_i$ such that at any iteration of the algorithm for each label in the set there is a plan that produces type i with that label. The algorithm proceeds in iterations attempting to apply each operator and update $\{\mathcal{F}_i\}$, and terminates at a state $\{\mathcal{F}_i^*\}$ when none of the operators can create a new label, or when for every goal type the corresponding set of possible labels contains a label dominated by the user's label:

$$\forall i \in \mathcal{T}(g) \quad \exists \ell \in \mathcal{F}_i^* : \quad \ell \prec \ell_{USER} \tag{21}$$

When new labels are created for types which can be downgraded, the downgraders are applied automatically before updating the corresponding set $\mathcal{F}_i$. Then the following invariant will be preserved at every iteration of the algorithm:

$$\forall i = 1, \ldots, n, \quad \forall \ell \in \mathcal{F}_i \Rightarrow \quad \ell \prec \delta_i \tag{22}$$

For clarity of presentation in our description of the algorithm we omit the bookkeeping necessary for restoring and returning the plan after the algorithm successfully terminates. The bookkeeping results in a trivial modification to the algorithm, which will require storing a record of the last operator together with the information about the labels used as inputs at every iteration when $\mathcal{F}_i$ is updated. This modification does not change the complexity of the algorithm, since a record of constant size is stored at every iteration.

$\mathbf{DP}(\mathbf{x^0}, \ell^\mathbf{0})$:

1 Let $\mathcal{F}_i := \begin{cases} \{\ell_i^0\}, & \text{for } i \in \mathcal{T}(x^0), \\ \emptyset, & \text{otherwise.} \end{cases}$

2 **for each** $\theta \in O : \forall i \in \mathcal{T}(p(\theta)) \Rightarrow \mathcal{F}_i \neq \emptyset$ **do**

3 **for each** $\ell \in \mathbb{L}$: $\forall i \in \mathcal{T}(p(\theta)) \Rightarrow \exists \ell' \in \mathcal{F}_i : \ell' \prec \ell$ **do**

4 **for each** $j \in \mathcal{T}(a(\theta))$ **do**

5 $\hat{\ell} := \ell \wedge \delta_j$;

6 **if** $\hat{\ell} \notin \mathcal{F}_j$ **then** $\mathcal{F}_j := \mathcal{F}_j \cup \{\hat{\ell}\}$;

7 **od**

8 **od**

9 **od**

10 **repeat 2** while (21) is not satisfied by $\{\mathcal{F}_i\}$ and the last iteration has updated any $\mathcal{F}_i$ in step 6.

The correctness of this algorithm follows from the observation that it will repeatedly evaluate every operator to create every possible label for every possible type, until the solution is found, or until all choices have been exhausted and no new labels can be created. The algorithm performs at most $\mathcal{O}(|\mathbb{L}|m)$ iterations of the *repeat* loop at line 10, since it terminates when no elements can be added to any $\mathcal{F}_i$, each of the sets $\mathcal{F}_i$ can contain at most $|\mathbb{L}|$ elements, and i in $\mathcal{F}_i$ is in the range $1 \leq i \leq n$, where the number of types $n \leq \mathcal{O}(m)$. At each iteration, the body of the *for* loop at line 2 is repeated $\mathcal{O}(m)$ times, the *for* loop at line 3 is repeated $\mathcal{O}(|\mathbb{L}|)$ times, and the loop at line 4 is repeated $\mathcal{O}(1)$ times, since we assume that $|\mathcal{T}(a(\theta))| \leq C$, where C is a constant. Note that condition (6) translates here exactly in (21), therefore Proposition 9 holds.

D.3 Proof of Proposition 10

Proposition 10. *The number of operations required to compose a workflow satisfying Bell-LaPadula access policy with downgraders does not exceed $\mathcal{O}(2^{2^{b+1}}|\mathcal{H}|^2m^2)$, where b is the cardinality of the set of*

distinct downgrader labels:

$$b := |\Delta| = |\{\delta_i, i \in \{1, \ldots, n\}\}|.$$

Proof.

This result is achieved by applying $\mathbf{DP}$ after a simple transformation of the labels to bit vectors of size 2^b, where each bit corresponds to a unique subset of Δ. The initial label $\ell_i^0 = (h_i, c_i)$ for each type i is mapped to a bit vector which describes the subsets of downgraders that must be applied to make the label accessible for $\ell_{USER} = (h_{user}, c_{user})$. To perform the mapping, for each category c such that $c \in c_i$ and $c \notin c_{user}$ the bit corresponding to the subset of labels in Δ that contain category c must be set to 1, and all other bits must be set to 0. The downgrader labels are mapped to bit vectors that set to 0 the bits for every subset of Δ in which the downgrader label participates, and have all other bits equal to 1. Under this mapping the goal is to construct a label vector equal to 0. It is easy to see that the workflow satisfying this requirement will be accessible for ℓ_{USER}, when mapped back to the original labels, and therefore $\mathbf{DP}$ will compose a valid workflow in at most $\mathcal{O}(2^{2^{b+1}}|\mathcal{H}|^2 m^2)$ operations, where $2^{2^{b+1}}$ is the square of the number of labels in the lattice after the transformation.

□