An Algebraic Approach to the Analysis of Constrained Workflow Systems

Jason Crampton
Information Security Group, Royal Holloway, University of London

7th June 2004

Abstract

The enforcement of authorization constraints such as separation of duty in workflow systems is an important area of current research in computer security. We briefly summarize our model for constrained workflow systems and develop a systematic algebraic method for combining constraints and authorization information. We then show how the closure of a set of constraints and the use of linear extensions can be used to develop an algorithm for computing authorized users in a constrained workflow system. We show how this algorithm can be used as the basis for a reference monitor. We discuss the computational complexity of implementing such a reference monitor and briefly compare our methods with the best existing approach.

Keywords Workflow specification, entailment constraints, linear extensions, satisfiability

1 Introduction

A workflow is a representation of an organizational or business process and is typically specified as a set of tasks and a set of dependencies between the tasks. Dependencies may include authorization constraints such as separation of duty requirements, where two different users must execute two different tasks. There exist several schemes and models in the literature for specifying separation of duty constraints [1, 2, 3, 5, 10, 12] and cardinality constraints [2] in computerized workflow systems.

These schemes are often based on a particular computational model: examples include logic programs [2, 12], active databases [5] and petri nets [1]. We introduced a simple specification scheme for authorization constraints that is independent of an underlying computational model and showed that it could be used to articulate *inter alia* separation of duty constraints and cardinality constraints [6]. In this paper, we exploit the simplicity and uniformity of our scheme to analyze the satisfiability of constrained workflow systems. We also show how this analysis can be used as the basis for a reference monitor for constrained workflow management systems and compare the computational complexity of our approach with that of Bertino *et al* [2], the most sophisticated existing approach in this area.

In the next section we review our work on modelling workflows and authorization constraints in workflows. Most importantly, we introduce entailment constraints, linear extensions of a workflow specification and methods for combining entailment constraints and authorization information. In Section 3 we introduce the concepts of *satisfiability* in a workflow and the *closure* of a set of entailment constraints. This leads naturally to the development of an algorithm for determining the set of users that are authorized to perform a task and who satisfy the entailment constraints that apply to that task.

In Section 4 we briefly describe a reference monitor for workflow systems, using this algorithm as the basis for deciding whether an access request should be granted. Finally we discuss future work.

2 A model for constrained workflows

A *workflow specification* is a partially ordered set of tasks T; if $\mathtt{t} < \mathtt{t}'$ then $\mathtt{t}$ must be performed before $\mathtt{t}'$ in any instance of the workflow. Let U be a set of users. A *workflow authorization schema* is a pair (T, A), where $A \subseteq \mathsf{T} \times U$ and $(\mathtt{t}, u) \in A$ means that u is *authorized to perform* (or *execute*) $\mathtt{t}$. (Generally, A will not encode task-user pairs directly; often such authorizations will be inferred from the assignment of tasks and users to common roles.)

Let $Rel(U)$ denote the set of all binary relations on U. (In other words, $Rel(U)$ is the powerset of $U \times U$.) Define

$$0' = \{(u, v) : u, v \in U, u \neq v\} \qquad 1' = \{(u, u) : u \in U\}$$
$$0 = \emptyset \qquad 1 = 1' \cup 0' = U \times U$$

An *entailment constraint* has the form $(D, (\mathtt{t}, \mathtt{t}'), \rho)$, where $D \subseteq U$, $\rho \in Rel(U)$ and $\mathtt{t} \not\geqslant \mathtt{t}'$. A *constrained workflow authorization schema* is a triple (T, A, C), where C is a set of entailment constraints.

Informally, if users u and u' perform $\mathtt{t}$ and $\mathtt{t}'$, respectively, and $u \in D$, then constraint $(D, (\mathtt{t}, \mathtt{t}'), \rho)$ is satisfied iff $(u, u') \in \rho$. (In other words, the constraint is not applied if $u \notin D$. We refer to D as the *domain* of the constraint.) Hence a separation of duty constraint can be expressed as $(U, (\mathtt{t}, \mathtt{t}'), 0')$ and a binding of duty constraint can be expressed as $(U, (\mathtt{t}, \mathtt{t}'), 1')$.

In fact, any binary relation between users can be used (including those that can be derived from contextual information). Hence it is possible to articulate constraints of the form "tasks $\mathtt{t}$ and $\mathtt{t}'$ must be performed by two different users in the same department". If we assume the existence of group-based or role-based authorization structures, then it is possible to induce an ordering (binary relation) on the set of users determined by the relative seniority of the roles to which each user is assigned. The relation $\ell \in Rel(U)$ will be used to denote an ordering on the set of users, which may be derived, depending on context, from role information, organizational information or the user groups to which users belong. We anticipate that this sort of relation will prove particularly important, because it is natural to implement access control in workflow systems using role-based techniques.

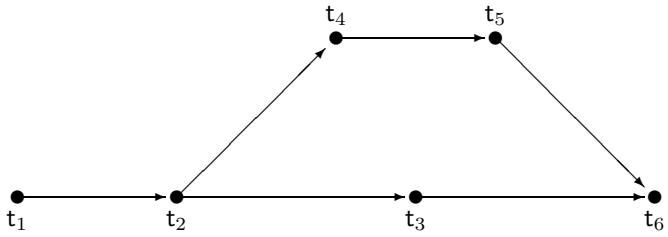
We have previously shown that cardinality constraints can be expressed as entailment constraints and that role-based authorization constraints should either be expressed as constraints on the authorization information or as entailment constraints based on the relative seniority of users (encoded by the ℓ relation) [6]. In that paper we also provide an example of a constrained workflow authorization schema; lack of space prevents us reproducing the example here.

2.1 Linear extensions and execution schedules

Let $\langle X, \leqslant \rangle$ be a partially ordered set. A *linear extension* of X is a total ordering of the elements of X that respects the ordering of the elements in X. In other words, $\langle X, \preccurlyeq \rangle$ is a linear extension of $\langle X, \leqslant \rangle$ if for all $x_1, x_2 \in X$, either $x_1 \preccurlyeq x_2$ or $x_2 \preccurlyeq x_1$, and if $x_1 \leqslant x_2$ then $x_1 \preccurlyeq x_2$. We denote the set of linear extensions of X by $\mathcal{L}(X)$.

Linear extensions are important in the context of workflows because they "linearize" a partially or-

dered set of tasks.[1] In other words, a linear extension of T represents a possible sequence of execution of the tasks in a workflow. Figure 1 shows a simple example of a workflow specification and its three linear extensions.



(a)

$\mathtt{t}_1 \prec \mathtt{t}_2 \prec \mathtt{t}_3 \prec \mathtt{t}_4 \prec \mathtt{t}_5 \prec \mathtt{t}_6$

$\mathtt{t}_1 \prec \mathtt{t}_2 \prec \mathtt{t}_4 \prec \mathtt{t}_3 \prec \mathtt{t}_5 \prec \mathtt{t}_6$

$\mathtt{t}_1 \prec \mathtt{t}_2 \prec \mathtt{t}_4 \prec \mathtt{t}_5 \prec \mathtt{t}_3 \prec \mathtt{t}_6$

(b)

Figure 1: A simple workflow specification and its linear extensions

Definition 1 *Let (T, A, C) be a constrained workflow authorization schema. An* execution schedule *for (T, A, C) is a pair (L, α), where $L \in \mathcal{L}(\mathsf{T})$ and $\alpha : L \to U$ assigns tasks to users, such that for all $\mathtt{t} \in \mathsf{T}$, $(\mathtt{t}, \alpha(\mathtt{t})) \in A$, and for all $(D, (\mathtt{t}, \mathtt{t}'), \rho) \in C$, $\alpha(\mathtt{t}) \in D$ implies $(\alpha(\mathtt{t}), \alpha(\mathtt{t}')) \in \rho$.*

In other words, an execution schedule respects the relative ordering of tasks in the workflow specification (since it is a linear extension of T), every task is performed by an appropriately authorized user and every entailment constraint is satisfied. A constrained workflow authorization schema is *satisfiable* if there exists an execution schedule for the schema (and *unsatisfiable* otherwise).

In general, the set of linear extensions in T can be generated in time $\mathcal{O}(|\mathcal{L}(\mathsf{T})|)$ [11] and computing $|\mathcal{L}(\mathsf{T})|$ is #P-complete [4]. However, if the width of the poset is small (as will be the case for a typical workflow specification), then the set of linear extensions can be computed quickly using dynamic programming techniques. We now discuss this is in more detail.

Proposition 2 *Suppose $\langle X, \leqslant \rangle$ is a poset and $x_1 \prec x_2 \prec \cdots \prec x_n$ is a linear extension of X. Then $\{x_1, \ldots, x_k\}$ is an order ideal in X, $1 \leqslant k \leqslant n$.*

Proof Suppose $\{x_1, \ldots, x_k\}$ is not an order ideal. Then there exists $y \in X$ such that $y \leqslant x_j$ for some j, $1 \leqslant j \leqslant k$, and $y \notin \{x_1, \ldots, x_k\}$. Therefore $y \leqslant x_j$ and $x_j \prec y$; hence $\{x_1, \ldots, x_k\}$ is not a linear extension and the result follows by contradiction. ■

Hence each linear extension is a directed path of maximal length in the graph of $\mathcal{I}(\mathsf{T})$, the lattice of order ideals of T.

Lemma 3 *Let $\langle X, \leqslant \rangle$ be a poset and let $\mathcal{I}(X)$ denote the set of order ideals in X. Then*

$$|\mathcal{I}(X)| \leqslant \left(\left\lceil \frac{|X|}{w} \right\rceil + 1 \right)^w,$$

where w is the width of X.

[1] We note that in certain circumstances, it will be possible for certain tasks in a workflow to execute in parallel. Specifically, if $\mathtt{t}$ and $\mathtt{t}'$ are tasks with $\mathtt{t} \parallel \mathtt{t}'$ and neither $\mathtt{t}$ nor $\mathtt{t}'$ appears in any constraint, then they may be executed in parallel. Such situations are outside the scope of this paper.

Proof By Dilworth's theorem [8], we can partition the poset $\langle X, \leqslant \rangle$ into w disjoint chains $C_1, \ldots, C_w$. Consider the poset $\langle X, \trianglelefteq \rangle$, where $x \trianglelefteq y$ iff $x, y \in C_i$ for some i and $x \leqslant y$ (in X). Then any order ideal in $\langle X, \leqslant \rangle$ is an order ideal in $\langle X, \trianglelefteq \rangle$. To see this, note that there is an isomorphism between the set of order ideals and the set of antichains, where an order ideal is mapped to the antichain comprising the maximal elements in the order ideal [7]. It is clear that any antichain in $\langle X, \leqslant \rangle$ must also be an antichain in $\langle X, \trianglelefteq \rangle$ by construction. In other words, the number of antichains in $\langle X, \leqslant \rangle$ (and hence the number of order ideals) is bounded by the number of antichains in $\langle X, \trianglelefteq \rangle$.

The number of antichains in $\langle X, \trianglelefteq \rangle$ is equal to $\prod_{i=1}^{w}(|C_i| + 1)$ (because we can choose at most one element from each chain in $\langle X, \trianglelefteq \rangle$). It is easy to show using elementary calculus that the product xy, subject to $x + y = k$, is maximized when $x = y = k/2$. Generalizing this result, we obtain

$$\prod_{i=1}^{w}(|C_i| + 1) \leqslant \prod_{i=1}^{w}\left(\left\lceil \frac{|X|}{w} \right\rceil + 1\right).$$

The result follows. ■

Hence the number of order ideals in T is bounded by $\left(\left\lceil \frac{|\mathsf{T}|}{w} \right\rceil + 1\right)^w$, where w is the width of T, and the directed paths can be computed using a breadth-first search whose complexity is linear in the number of nodes of the graph. In other words, if w is small, the number of order ideals can be computed in time polynomial in the number of tasks in the workflow specification.

Figure 2 shows the lattice of order ideals and the lattice of antichains for the workflow specification depicted in Figure 1. (We have adopted the usual convention in Hasse diagrams that $x \leqslant y$ implies y is above x in the diagram.) In our example $w = 2$ and the number of order ideals is 9.

2.2 The algebra of entailment constraints

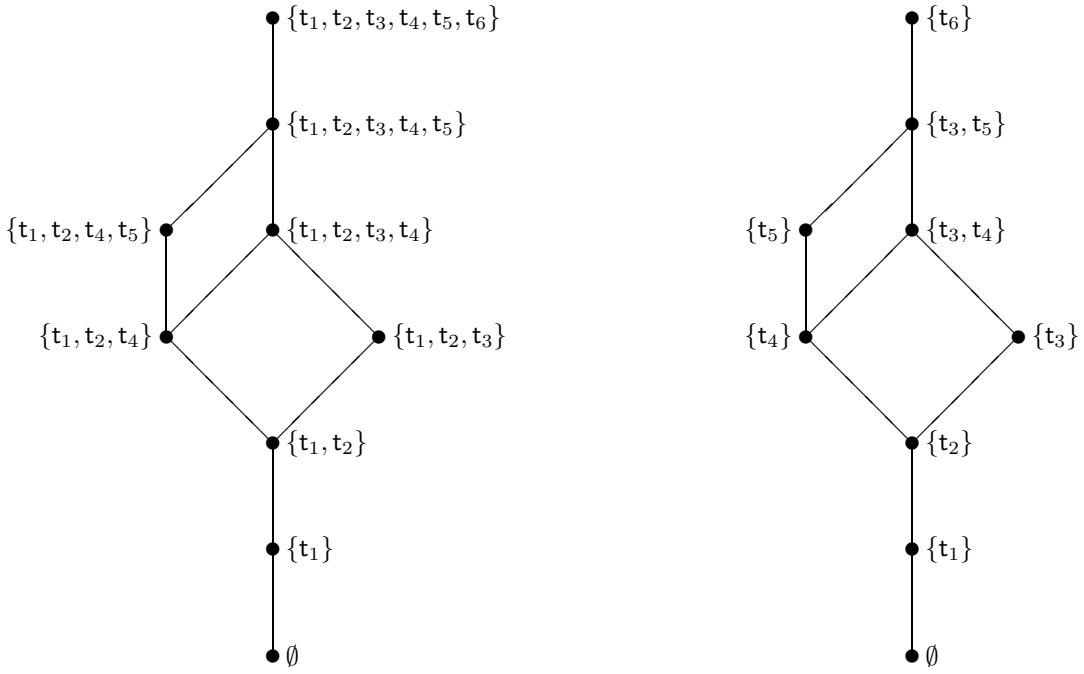
In this section we state without proof a number of simple results concerning entailment constraints. The reader is referred to our earlier work for further details [6]. We conclude the section with a new result that enables us to omit authorization information from a workflow schema, thereby facilitating the analysis of workflow systems.

Proposition 4 (Merging domains) *Let* $(\mathsf{T}, A, \{(D_1, (\mathsf{t}, \mathsf{t}'), \rho), (D_2, (\mathsf{t}, \mathsf{t}'), \rho)\}$ *be a constrained workflow authorization schema. Then* (L, α) *is a workflow execution schedule for* $(\mathsf{T}, A, \{(D_1, (\mathsf{t}, \mathsf{t}'), \rho), (D_2, (\mathsf{t}, \mathsf{t}'), \rho)\}$ *iff* (L, α) *is a workflow execution schedule for* $(\mathsf{T}, A, \{D_1 \cup D_2, (\mathsf{t}, \mathsf{t}'), \rho\})$.

Proposition 5 (Expanding the domain) *Let* $(\mathsf{T}, A, \{(D, (\mathsf{t}, \mathsf{t}'), \rho)\})$ *be a constrained workflow authorization schema and define* $\sigma = (U \setminus D) \times U$. *Then* (L, α) *is a workflow execution schedule for* $(\mathsf{T}, A, \{(D, (\mathsf{t}, \mathsf{t}'), \rho)\})$ *iff* (L, α) *is a workflow execution schedule for* $(\mathsf{T}, A, \{(U, (\mathsf{t}, \mathsf{t}'), \rho \cup \sigma)\})$.

Proposition 6 (Merging constraints) *Let* $W = (\mathsf{T}, A, \{((\mathsf{t}, \mathsf{t}'), \rho_1), ((\mathsf{t}, \mathsf{t}'), \rho_2)\})$ *be a constrained workflow authorization schema. Then* (L, α) *is an execution schedule for* W *iff* (L, α) *is an execution schedule for* $(\mathsf{T}, A, \{((\mathsf{t}, \mathsf{t}'), \rho_1 \cap \rho_2)\})$.

In other words, we can assume that the domain of every constraint is U (by Proposition 5), and that for each pair of tasks $(\mathsf{t}, \mathsf{t}')$ there is a single constraint (by Proposition 6).



(a) Order ideals

(b) Antichains

Figure 2: Lattices derived from the workflow specification in Figure 1

Proposition 7 (Composing constraints) *Let* $W = (\mathsf{T}, A, \{((\mathsf{t}, \mathsf{t}'), \rho_1), ((\mathsf{t}', \mathsf{t}''), \rho_2)\})$ *be a constrained workflow authorization schema. Then* (L, α) *is an execution schedule for* W *iff* (L, α) *is an execution schedule for* $(\mathsf{T}, A, \{((\mathsf{t}, \mathsf{t}'), \rho_1), ((\mathsf{t}', \mathsf{t}''), \rho_2), ((\mathsf{t}, \mathsf{t}''), \rho_1\rho_2)\})$*, where*

$$\rho_1\rho_2 = \{(u, w) : \exists v \in U, (u, v) \in \rho_1, (v, w) \in \rho_2\}.$$

It is important to note that if (L, α) is an execution schedule for $(\mathsf{T}, A, \{((\mathsf{t}, \mathsf{t}''), \rho_1\rho_2)\})$, then it is not necessarily an execution schedule for $(\mathsf{T}, A, \{((\mathsf{t}, \mathsf{t}'), \rho_1), ((\mathsf{t}', \mathsf{t}''), \rho_2)\})$. (Although we have $(\alpha(\mathsf{t}), \alpha(t'')) \in \rho_1\rho_2$, we can not necessarily infer that there exists an authorized user for t'.) In other words, we cannot delete the constraints from which a compound constraint is derived. Unfortunately, the composition of relations is neither commutative nor associative. However, for each linear extension of the workflow there is a unique order in which the relations are composed.

At the moment we only consider entailment constraints to be specifications of security policy requirements such as separation of duty. However, we can view the authorization information as a set of entailment constraints on the execution of tasks. In particular, let $\mathsf{T} = \{\mathsf{t}_1, \ldots, \mathsf{t}_n\}$ and for all $\mathsf{t}_i \not\geqslant \mathsf{t}_j$, define $a_{ij} = U(\mathsf{t}_i) \times U(\mathsf{t}_j)$, where $U(\mathsf{t}) = \{u \in U : (\mathsf{t}, u) \in A\}$. Then the entailment constraint $((\mathsf{t}_i, \mathsf{t}_j), a_{ij})$, is only satisfied if two appropriately authorized users perform tasks t_i and t_j. If there exists an entailment constraint of the form $((\mathsf{t}_i, \mathsf{t}_j), \rho_{ij})$ then we form the new constraint $((\mathsf{t}_i, \mathsf{t}_j), \rho_{ij} \cap a_{ij})$. More formally, we have the following result. The proof of this result follows immediately from the definition of an execution schedule and is omitted.

Proposition 8 (Incorporating authorization information) *Let* $\mathsf{W} = (\mathsf{T}, A, \{((\mathsf{t}, \mathsf{t}'), \rho)\})$ *be a constrained workflow authorization schema. Then* (L, α) *is an execution schedule for* W *iff* (L, α) *is an*

execution schedule for $(\mathsf{T}, \mathsf{T} \times U, \{((\mathsf{t}, \mathsf{t}'), \rho \cap a)\})$*, where* $a = \{(u, u') : (\mathsf{t}, u), (\mathsf{t}', u') \in A\}$.

In other words, we can express all the information required to make an authorization decision in terms of entailment constraints. Hence it is sufficient from a theoretical point of view to consider workflow schemata of the form (T, C), although, from a practical perspective, it is clearly more natural to include authorization information.

3 Satisfiability in workflow systems

There are three questions that are of interest:

- Is a constrained workflow authorization schema satisfiable? Given a constrained workflow authorization schema (T, A, C), is it possible for some instance of the workflow to complete. In other words, is there an assignment of tasks to users $\alpha : \mathsf{T} \to U$ and a linear extension $L \in \mathcal{L}(\mathsf{T})$ such that (L, α) is an execution schedule.
- Is an instance of a workflow schema satisfiable? We write t to denote an instance of the task $\mathsf{t} \in \mathsf{T}$. In other words, given a constrained workflow authorization schema (T, A, C) and an order ideal $I \subseteq \mathsf{T}$, where $t_i \in I$ has been executed by u_i, is it possible to extend the ideal to a linear extension of T and to find an assignment of the remaining tasks to users such that the resulting linear extension and the assignment of tasks to users forms an execution schedule for (T, A, C).
- Given a satisfiable workflow authorization schema, is it possible to design a reference monitor so that every instance of that schema is satisfiable? In other words, is it possible to design a decision process that only permits a request from a user to execute a task if the remaining tasks can be completed. Clearly, an answer to the previous question will provide a blueprint for the design of such a reference monitor.

3.1 The closure of a set of constraints

Let $\mathsf{W} = (\mathsf{T}, C)$ be a constrained workflow authorization schema. We assume that $c_{ij} = ((\mathsf{t}_i, \mathsf{t}_j), \rho_{ij})$ is defined whenever $\mathsf{t}_i \not\geqslant \mathsf{t}_j$ (setting $\rho_{ij} = 1$ where necessary) and define

$$C^2 = C \cup \{((\mathsf{t}, \mathsf{t}''), \rho_1\rho_2) : ((\mathsf{t}, \mathsf{t}'), \rho_1), ((\mathsf{t}', \mathsf{t}''), \rho_2) \in C\},$$
$$C^k = C^{k-1} \cup \{((\mathsf{t}, \mathsf{t}''), \rho_1\rho_2) : ((\mathsf{t}, \mathsf{t}'), \rho_1) \in C^{k-1}, ((\mathsf{t}', \mathsf{t}''), \rho_2) \in C\}.$$

Let $|\mathsf{T}| = n$. Then C^{n-1} denotes the set of all possible constraints that can be derived from the original set of constraints C using composition. Let $c^k_{ij} \in C^k$ denote the constraint $((\mathsf{t}_i, \mathsf{t}_j), \rho^k_{ij})$.[2] Define the *closure* of C, denoted C^*, to be the set of constraints $c^*_{ij} = ((\mathsf{t}_i, \mathsf{t}_j), \rho^*_{ij})$, where $\mathsf{t}_i \not\geqslant \mathsf{t}_j$ and $\rho^*_{ij} = \bigcap_{k=1}^{n-1} \rho^k_{ij}$, $1 \leqslant i, j \leqslant n$. If a pair of users satisfy a constraint $((\mathsf{t}, \mathsf{t}'), \rho) \in C^*$, then for every linear extension of T, there exists a sequence of users that can execute the sequence of tasks between t and t'.

Informally, we can regard C as a labelled directed graph in which the set of nodes is T and an edge $(\mathsf{t}_i, \mathsf{t}_j)$ labelled ρ_{ij} exists if $\mathsf{t}_i \not\geqslant \mathsf{t}_j$. The constraints in C^k are the paths of length k in this graph. In fact, we can realize C as a matrix (in which the ijth entry is ρ_{ij}) and "multiply" the matrix by itself $n - 1$ times to derive $C^2, \ldots, C^{n-1}$.

[2] There may be more than one constraint that can be derived for tasks t_i and t_j. In this case we simply take the intersection of the relations for each of these constraints to derive a single constraint.

Theorem 9 *Let* (T, C) *be a constrained workflow authorization schema and let* $\mathtt{t} \in \mathsf{T}$ *be a minimal element and* $\mathtt{t}' \in \mathsf{T}$ *be a maximal element. Then* $((\mathtt{t}, \mathtt{t}'), \rho) \in C^*$ *for some* $\rho \subseteq 1$ *and* (T, C) *is satisfiable if* $\rho \neq \emptyset$*.*

Proof sketch There exists a "path" of length $n-1$ between $\mathtt{t}$ and $\mathtt{t}'$ and hence there exists a constraint of the form $((\mathtt{t}, \mathtt{t}'), \rho)$. A simple induction (using Propositions 6 and 7 for the base case) shows that every constraint on that path is satisfied. Hence if ρ is non-empty, then there exists a pair of users that can perform the first task and last task and a sequence of users that satisfies all the constraints in between. In other words, there exists an execution schedule for (T, C). ■

Unfortunately, it is rather difficult to compute the closure of C in this way directly because the graph of C is not acyclic.[3] More specifically, we need to be able to distinguish between constraints that arise because of paths (in which each node is visited at most once) and those that arise because of walks (in which a node may be visited more than once). One possible way of doing this is to compute the length of the longest path between each pair of tasks and omit any constraints that arise due to a walk between a given pair of tasks which exceed this length. The computation of the longest path between a pair of nodes in a directed graph is NP-complete [9, Problem ND29].

Alternatively, given a workflow schema $\mathsf{W} = (\mathsf{T}, C)$, we can enumerate all possible linear extensions, thereby creating a family of workflow schemata $\mathcal{W}$ in which each specification is a totally ordered set of tasks. For each such schema we compute the closure of the set of constraints. Finally, we can create a single workflow schema $\mathsf{W}^* = (\mathsf{T}, C^*)$, where for all $((\mathtt{t}_i, \mathtt{t}_j), \rho_{ij} \in C^*$, ρ_{ij} is obtained by taking the intersection of the relations in every constraint of the form $((\mathtt{t}_i, \mathtt{t}_j), \rho)$ in $\mathcal{W}$.

3.2 An algorithm for computing execution schedules

Figure 3 illustrates an algorithm (written in pseudo-code) that computes $V(\mathtt{t}, \mathtt{t}')$ for each pair $(\mathtt{t}, \mathtt{t}')$, where $V(\mathtt{t}, \mathtt{t}')$ is the set of users that can execute $\mathtt{t}$ and $\mathtt{t}'$ (in that order) given the authorization information and the entailment constraints in the schema that apply to $\mathtt{t}$ and $\mathtt{t}'$. The basic strategy is to initialize each $V(\mathtt{t})$ to the set of users that are authorized to perform T (line `02`) and then, for each linear extension, to apply all the possible constraints (including those derived from authorization information) (lines `07`–`08`). Essentially, the algorithm is applying Proposition 8 and computing a new relation for each entailment constraint. If one of these relations is empty, then the algorithm terminates prematurely (line `08`), since there does not exists a pair of authorized users that comply with the entailment constraints. Finally, for each task $\mathtt{t}$ we (re-)compute the set of users that can perform $\mathtt{t}$ (lines `10`–`11`).

The overall time complexity of the algorithm is $\mathcal{O}(|T|^w|T|^2|U|^4) = \mathcal{O}(|T|^{w+2}|U|^4)$, since the number of linear extensions is $\mathcal{O}(|T|^w)$ (see Section 2.1), the number of constraints is $\mathcal{O}(|T|^2)$ and the comparison in line `07` is $\mathcal{O}(|U|^4)$ in the worst case. Note that the computational complexity of the comparison in line `07` dominates the complexity of the computations required in lines `10` and `11`, which are $\mathcal{O}(|U|^2)$. Note also that if R is $0'$ or $1'$, then the computation in line `07` is a simple comparison of $V(i)$ and $V(j)$ and hence has time complexity $\mathcal{O}(|U|^2)$. In other words, if we restrict our attention to cardinality, separation of duty and binding of duty constraints, then the overall complexity reduces to $\mathcal{O}(|\mathsf{T}|^{w+2}|U|^2)$. (Recall that cardinality constraints can be modelled using separation of duty constraints [6].)

[3] For example, if we define the constraints $((\mathtt{t}_3, \mathtt{t}_4), 0')$ and $(\mathtt{t}_4, \mathtt{t}_3), 0')$ for the workflow in Figure 1, meaning that the same user cannot perform both $\mathtt{t}_3$ and $\mathtt{t}_4$, then we have a cycle of length 2 in the graph.

```
01    for i = 1 to |T|
02      let V(i) = set of users authorized to perform task i
03    for each linear extension
04      for i = 1 to |T|
05        for j = 1 to |T|
06          if ((i,j),R) ∈ C
07            let V(i,j) = (V(i) × V(j)) ∩ R
08            if V(i,j) is empty then exit
09            else
10              let V(i) = set of users in first position of V(i,j)
11              let V(j) = set of users in second position of V(i,j)
```

Figure 3: An algorithm for determining whether an execution schedule exists

4 A reference monitor for constrained workflows

A *workflow system* is a pair $\mathcal{S} = (\mathcal{W}, \mathcal{M})$, where $\mathcal{W}$ is a set of workflow schemata and $\mathcal{M}$ is a reference monitor. A *reference monitor* is an abstract machine for deciding whether an access request from a user will be granted. A workflow *instance* is created (instantiated) when the first task in some linear extension of T is executed.

Let $\mathsf{W} = (\mathsf{T}, A, C)$ be a constrained workflow authorization schema. Then we denote an instance of this schema by W and an instance of task $\mathtt{t}$ by t. A workflow instance completes if every task in the workflow specification is performed by some user.

We will say that a workflow system is *complete* if every instance of every schema is guaranteed to complete. A workflow instance, in general, will not complete because the execution of certain tasks by certain users and the existence of entailment constraints in the schema may restrict the users that can perform subsequent tasks. Hence, a workflow system will be complete only if the reference monitor is able to identify and deny tasks that would prevent subsequent tasks from being executed because certain entailment constraints could not be satisfied. However, a reference monitor that guarantees a workflow system is complete is likely to be computationally expensive [2].

In the context of workflow systems, a user requests the permission to execute a task in a workflow instance. In other words, $\mathcal{M}$ is a function that takes a triple (t, i, u) and returns `allow` if the request is granted and `deny` otherwise. The triple (t, i, u) is interpreted as a request by user u to execute task $\mathtt{t} \in \mathsf{T}$ in W_i, the ith instance of $\mathsf{W} = (\mathsf{T}, A, C)$.

Let W be a workflow instance in which all the tasks in $\mathsf{T}' \subseteq \mathsf{T}$ have been executed, where T' is an order ideal in T. Then this workflow instance can be represented as a function $I : \mathsf{T}' \to U$, where user $I(\mathtt{t})$ performed task $\mathtt{t}$. The execution of a workflow instance is constrained by I and C. Given a workflow authorization schema (T, A, C), let $\mathsf{W}|I$ denote the workflow schema $(\mathsf{W}, A|I, C)$, where

$$A|I = \{(\mathtt{t}, I(\mathtt{t})) : \mathtt{t} \in \mathsf{T}'\} \cup \{(\mathtt{t}, u) \in A : \mathtt{t} \in \mathsf{T} \setminus \mathsf{T}'\}.$$

In other words, $\mathsf{W}|I$ is a constrained workflow authorization schema in which each task $\mathtt{t} \in \mathsf{T}'$ has a single authorized user $I(\mathtt{t})$; that is, the user that performed $\mathtt{t}$ in instance I.

In general, given a partially completed workflow instance I and a request by u to execute $\mathtt{t}$ in this instance there are three questions a reference monitor could consider:

Q1 Is u authorized to perform $\mathtt{t}$?

Q2 Are all constraints in which $\mathtt{t}$ is the consequent task satisfied?

Q3 Can the workflow complete if u performs t?

The reference monitor must certainly guarantee that the answers to the first two questions are yes. It is up to the designers of the reference monitor to decide whether the third question should always have an affirmative answer. Indeed, some research has been done on overriding (that is, not enforcing) constraints in the event that a workflow cannot complete because of previous task executions and the existence of constraints [12]. We say a reference monitor is *enforcement compliant* if it guarantees (for all requests) that the answers to the first two questions are yes and *completion compliant* if it guarantees that the answer to each of the three questions is yes.

Let $\mathsf{W} = (\mathsf{T}, A, C)$ be a constrained workflow authorization schema. In order to implement a reference monitor for this workflow, we compute C^* and use the algorithm in Figure 3 to establish that an execution schedule for the workflow exists and to compute a relation $V \subseteq A \subseteq \mathsf{T} \times U$, where $(\mathsf{t}, u) \in V$ implies that u is authorized to perform t and all entailment constraints can be satisfied. We write $V(\mathsf{t})$ to denote the set $\{u \in U : (\mathsf{t}, u) \in V\}$.

Let us first consider the case where $\mathsf{t} \in \mathsf{T}$ is a minimal element (and hence can be the first task executed in a workflow). In this case, $I = \emptyset$ and $\mathsf{W}|I = \mathsf{W}$.[4] Then a request to execute t may be granted if $(\mathsf{t}, u) \in V$. If the request were to be granted, then $I = \{(\mathsf{t}, u)\}$; a completion compliant reference monitor must recalculate V for the workflow $\mathsf{W}|\{(\mathsf{t}, u)\}$. If $V(\mathsf{t}') = \emptyset$ for some $\mathsf{t}' \in \mathsf{T}$ then the workflow schema, and hence the workflow instance, cannot be satisfied. Hence, in order to implement a completion compliant reference monitor, we simply run the algorithm in Figure 3 for the workflow $\mathsf{W}|\{(\mathsf{t}, u)\}$ *before* granting the request (t, i, u). If the request is granted, the next request must be evaluated for the workflow $\mathsf{W}|\{(\mathsf{t}, u)\}$.

In the general case, let I be an instance of $\mathsf{W} = (\mathsf{T}, A, C)$ and let $I \cup \{\mathsf{t}\}$ be an order ideal in T. Then a request by u to execute t in this instance of W is granted by a completion compliant reference monitor if there exists an execution schedule for $\mathsf{W}|I$ such that u executes t and there exists an execution schedule for $\mathsf{W}|(I \cup \{(\mathsf{t}, u)\})$. In other words, we simply run the algorithm in Figure 3 for the workflow $\mathsf{W}|(I \cup \{(\mathsf{t}, u)\})$ *before* granting the request (t, i, u). If the request is granted, the next request must be evaluated for the workflow $\mathsf{W}|(I \cup \{\mathsf{t}, u\})$.

In summary, a completion compliant reference monitor must calculate V before any instance of the workflow is created. The reference monitor must also re-calculate V before every request (to check that the request is completion compliant) and update the workflow after every successful request (to ensure that the fact that a particular user executed a particular task is considered in enforcing constraints that apply to subsequent tasks).

The only comprehensive treatment of completion compliant workflow systems in the literature [2] includes an algorithm for "user planning", which associates tasks with a user-role pair. The complexity of this algorithm is $\mathcal{O}((N_R \cdot N_U \cdot N_{act})^{|\mathsf{T}|})$, where N_R is the maximum number of roles associated with any task in the workflow, N_U is the maximum number of users associated with any role in the workflow and N_{act} is the maximum number of activations associated with any task in the workflow. This algorithm is run before any workflow instance is created and, in the worst case, is also run when a request is received and when a task has been successfully executed. Our algorithm has time complexity $\mathcal{O}(|T|^{w+2} \cdot |N_U|^4)$. It has better time complexity than the user planner algorithm for several reasons:

- firstly, we only consider entailment constraints, which makes the analysis of constraints uniform and hence simpler;

[4] $I = \emptyset$ in the sense that there are no pairs $(\mathsf{t}, I(\mathsf{t}))$ defined.

- secondly, we only consider user-based constraints (having shown that role-based constraints can be enforced in other ways [6]);
- thirdly, we do not compute every possible sequence of tasks and users, instead computing the closure and determining if a workflow instance can complete (without explicitly calculating a sequence of tasks and users).

We note that the performance of the user planner algorithm can be improved by adopting certain heuristics, but the worst case complexity is still exponential in the number of tasks in the workflow specification.

5 Concluding remarks

The analysis of satisfiability in this paper suggests that there are distinct advantages to our approach to authorization constraints in workflow systems. We believe our approach has the following advantages over existing approaches:

- The ability to treat most, if not all, useful authorization constraints as special cases of entailment constraints means that the analysis of a set of authorization constraints for a workflow is greatly simplified.
- The ability to express authorization information in terms of entailment constraints means that satisfiability questions can be analyzed entirely in the context of the closure of a set of entailment constraints.
- The fact that our model for constraints is independent of any underlying computational model coupled with its simplicity means that it can be easily implemented in a variety of ways.

There are numerous opportunities for further research. Perhaps the most obvious of these is a prototype implementation, perhaps using an off-the-shelf relational database management system, to assess the usability and scalability of our approach.

In many workflow models, a task may be repeated several times within a workflow. When the number of occurrences of the task is fixed in each instance of the workflow, this can be modelled using cardinality constraints. However, in other models, the number of occurrences is allowed to vary. This clearly makes the analysis of satisfiability and the design of a reference monitor for such systems more complex. However, we believe that extending our model to include an entailment constraints of the form $((\mathtt{t},\mathtt{t}),\rho)$ may provide a suitable platform for investigating such workflow systems. The investigation of this topic will be one of our immediate priorities in future research.

Another avenue for further work is to consider substituting "proxy users" for actual users in the analysis of the workflow schema. The complexity of the algorithm is polynomial in the number of users and it is likely that the number of tasks will be considerably smaller than the number of users. A proxy user is simply identified with a subset of tasks (and hence is synonymous with a role). This means that proxy users can also be used to derive a role hierarchy for the workflow.

The number of proxy users is certainly bounded by $2^{|\mathsf{T}|}$, and we would expect the number of roles to be bounded by the number of tasks. However, we also need to consider "compound roles" consisting of tasks assigned to two or more roles. (For example, we might identify that $\{\mathtt{t}_1,\mathtt{t}_2\}$ naturally form one role and $\{\mathtt{t}_3,\mathtt{t}_4\}$ form another. We need to allow for the fact that a user may be assigned to both roles and so there must be a proxy user for the set $\{\mathtt{t}_1,\mathtt{t}_2,\mathtt{t}_3,\mathtt{t}_4\}$.) Hence, in general the number of proxy users will actually be bounded by the number of antichains in the role hierarchy. We would expect

that the number of roles is less than $|\mathsf{T}|$. Hence we can run the algorithm in Figure 3 using proxy users rather than actual users, thereby reducing the time complexity of the algorithm to $\mathcal{O}(|T|^{W+w+2})$, where W denotes the width of the role hierarchy.

Acknowledgements We would like to thank Frank Ruskey for his helpful comments on generating linear extensions.

References

[1] V. Atluri and W. Huang. An authorization model for workflows. In *Proceedings of the 4th European Symposium on Research in Computer Security*, pages 44–64, 1996.

[2] E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security*, 2(1):65–104, 1999.

[3] R.A. Botha and J.H.P. Eloff. Separation of duties for access control enforcement in workflow environments. *IBM Systems Journal*, 40(3):666–682, 2001.

[4] G. Brightwell and P. Winkler. Counting linear extensions. *Order*, 8:225–242, 1991.

[5] F. Casati, S. Castano, and M. Fugini. Managing workflow authorization constraints through active database technology. *Information Systems Frontiers*, 3(3):319–338, 2001. Technical Report HPL-2000-156, Hewlett Packard Laboratories.

[6] J. Crampton. On the satisfiability of authorization constraints in workflow systems. Technical Report RHUL–MA–2004–1, Department of Mathematics, Royal Holloway, University of London, 2004. `http://www.ma.rhul.ac.uk/techreports/`.

[7] J. Crampton and G. Loizou. The completion of a poset in a lattice of antichains. *International Mathematical Journal*, 1(3):223–238, 2001.

[8] R.P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51:161–6, 1950.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability*. W.H. Freeman and Company, San Francisco, California, 1979.

[10] K. Knorr and H. Stormer. Modeling and analyzing separation of duties in workflow environments. In *Trusted Information: The New Decade Challenge, IFIP TC11 Sixteenth Annual Working Conference on Information Security*, pages 199–212, 2001.

[11] G. Pruesse and F. Ruskey. Generating linear extensions fast. *SIAM Journal on Computing*, 23(2):373–386, 1994.

[12] J. Wainer, P. Barthelmess, and A. Kumar. W-RBAC – A workflow security model incorporating controlled overriding of constraints. *International Journal of Cooperative Information Systems*, 12(4):455–486, 2003.