On the Parameterized Complexity of the Workflow Satisfiability Problem

Jason Crampton
Royal Holloway
University of London
United Kingdom
jason.crampton@rhul.ac.uk

Gregory Gutin
Royal Holloway
University of London
United Kingdom
g.gutin@rhul.ac.uk

Anders Yeo
University of Johannesburg
South Africa
anders.yeo.work@gmail.com

ABSTRACT

A workflow specification defines a set of steps and the order in which those steps must be executed. Security requirements may impose constraints on which groups of users are permitted to perform subsets of those steps. A workflow specification is said to be satisfiable if there exists an assignment of users to workflow steps that satisfies all the constraints. An algorithm for determining whether such an assignment exists is important, both as a static analysis tool for workflow specifications, and for the construction of run-time reference monitors for workflow management systems. Finding such an assignment is a hard problem in general, but work by Wang and Li in 2010 using the theory of parameterized complexity suggests that efficient algorithms exist under reasonable assumptions about workflow specifications. In this paper, we improve the complexity bounds for the workflow satisfiability problem. We also generalize and extend the types of constraints that may be defined in a workflow specification and prove that the satisfiability problem remains fixed-parameter tractable for such constraints.

Categories and Subject Descriptors

D4.6 [**Operating Systems**]: Security and Protection—*Access controls*; F2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; H2.0 [**Database Management**]: General—*Security, integrity and protection*

General Terms

Algorithms, Security, Theory

Keywords

authorization constraints, workflow satisfiability, parameterized complexity

CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA.
Copyright 2012 ACM 978-1-4503-1651-4/12/10 ...$15.00.

1. INTRODUCTION

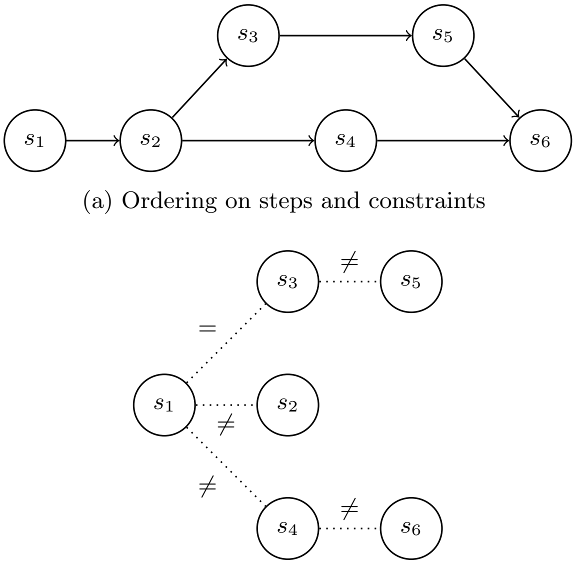
It is increasingly common for organizations to computerize their business and management processes. The coordination of the tasks or steps that comprise a computerized business process is managed by a workflow management system (or business process management system). Typically, the execution of these steps will be triggered by a human user, or a software agent acting under the control of a human user, and the execution of each step will be restricted to some set of authorized users.

A workflow typically specifies the tasks that comprise a business process and the order in which those tasks should be performed. Moreover, it is often the case that some form of access control should be applied to the execution of tasks. Hence, most workflow management systems may implement security controls that enforce authorization rules and business rules, in order to comply with statutory requirements or best practice. It is such "security-aware" workflows that will be the focus of the remainder of this paper. The most widely cited rules include separation-of-duty (also known as the "two-man" or "four-eyes" rule), which may be used to prevent sensitive combinations of steps being performed by a single user, and binding-of-duty, which requires that a particular combination of steps is executed by the same user.

A simple, illustrative example for purchase order processing [5] is shown in Figure 1. In the first step of the workflow, the purchase order is created and approved (and then dispatched to the supplier). The supplier will present an invoice, which is processed by the create payment step. When the supplier delivers the ordered goods, a goods received note (GRN) must be signed and countersigned. Only then may the payment be approved. Note that a workflow specification need not be linear: the processing of the GRN and of the invoice can occur in parallel, for example.

In addition to defining the order in which steps must be performed, the workflow specification includes rules to prevent fraudulent use of the purchase order processing system. These rules take the form of constraints on users that can perform pairs of steps in the workflow: the same user may not sign and countersign the GRN, for example.

It is apparent that it may be impossible to find an assignment of authorized users to workflow steps such that all constraints are satisfied. In this case, we say that the workflow specification is *unsatisfiable.* The Workflow Satisfiability Problem (WSP) is known to be NP-hard, even when the set of constraints only includes constraints that have a



(a) Ordering on steps and constraints

(b) Constraints

s_1	create purchase order
s_2	approve purchase order
s_3	sign goods received note
s_4	create payment
s_5	countersign goods received note
s_6	approve payment
$\neq$	different users must perform steps
$=$	same user must perform steps

(c) Legend

Figure 1: A simple constrained workflow for purchase order processing

relatively simple structure (and that would arise regularly in practice).[1]

An algorithm that solves WSP can be used to perform a static analysis of a workflow specification to determine whether there exists an instance of the workflow that can complete. The NP-hardness of the problem suggests that the worst-case run-time of such an algorithm will be exponential in the size of the input. Hence, it is important to find an algorithm that is as efficient as possible. Moreover, we may wish to construct a reference monitor that decides whether a user should be permitted to execute a particular step in a workflow instance. Part of that reference monitor's functionality will be to confirm that the user is authorized to execute that step; the second part of its functionality will be to determine whether allowing the request would render the remaining steps in the workflow instance unsatisfiable [5]. Assuming that such a reference monitor should incur as little delay as possible (in the interest of the end-user experience), it becomes even more important to find an algorithm that can decide WSP as efficiently as possible.

It has been argued that it would be of practical value to be able to define constraints in terms of organizational structures, rather than just the identity of particular users [17]. The first contribution of this paper is to introduce a model for hierarchical organizations based on the notion of equivalence classes and partition refinements. We demonstrate how to construct an instance of our model from a management structure and illustrate why constraints defined over such models are of practical value.

Wang and Li [17] observed that the number of steps in a workflow is likely to be small relative to the size of the input to the workflow satisfiability problem. This observation led them to study the problem using tools from parameterized complexity and to prove that the problem is fixed-parameter tractable for certain classes of constraints. These results demonstrate that it is feasible to solve WSP for many workflow specifications in practice. However, Wang and Li also showed that for many types of constraints the problem is fixed-parameter intractable unless one of the parameterized complexity hypotheses, FPT $\neq$ W[1], fails, which is highly unlikely. (We provide a short introduction to parameterized complexity in Section 4.1.)

In this paper, we extend the results of Wang and Li in three different ways.

1. First, we significantly improve their bound on the fixed-parameter complexity of WSP for those classes of constraints for which Wang and Li proved that the problem is tractable. This bound arises from a new approach to the problem and the application of a powerful, recent result in the area of exponential-time algorithms [3]. Moreover, we demonstrate that our result cannot be significantly improved, provided a well-known hypothesis about the complexity of solving 3-SAT holds.
2. Our second extension to the work of Wang and Li is to define constraints in terms of hierarchical structures and to prove that WSP remains fixed-parameter tractable in the presence of such hierarchical structures and hierarchy-related constraints.
3. In their work, Wang and Li impose a restriction on what we might informally call the structure of a constraint. Our final contribution is to remove this restriction and prove that Wang and Li's results on fixed-parameter tractability of WSP still hold, albeit with an increase in the complexity of the algorithm that solves the problem.

In the next section, we introduce the workflow satisfiability problem, as defined by Wang and Li. In Section 3, we introduce a model for an organizational hierarchy and a class of constraint relations defined in terms of such hierarchies. Then, in Section 4, we introduce our approach to the workflow satisfiability problem and prove a result that reduces the complexity of the problem for a particular class of constraints. We demonstrate that this approach generalizes to workflow specifications that include constraints defined over an organizational hierarchy and prove that the satisfiability problem remains fixed-parameter tractable. In Section 5, we discuss a number of extensions to the workflow satisfiability problem, including generalizations of the constraint structure and of the satisfiability problem itself. The paper concludes with a summary of our contributions and discussions of related and future work.

2. THE SATISFIABILITY PROBLEM

In this section, we introduce our notation and definitions, derived from earlier work by Crampton [5] and Wang and Li [17], and then define the workflow satisfiability problem.

[1] In particular, the Graph k-Colorability problem can be reduced to a special case of WSP in which the workflow specification only includes separation-of-duty constraints [17].

Let U be a set of users and let $Rel(U)$ denote the set of all binary relations on U. In other words, $Rel(U)$ is the powerset of $U \times U$. We will write $\rho_=$ and $\rho_{\neq}$ to denote the relations $\{(u,u) : u \in U\}$ and $\{(u,v) : u, v \in U, u \neq v\}$, respectively. Given $\rho \in Rel(U)$, let $\widetilde{\rho} = \{(v,u) : (u,v) \in \rho\}$. Note that if ρ is symmetric (as are $\rho_=$ and $\rho_{\neq}$), then $\rho = \widetilde{\rho}$. If $(X, \leqslant)$ is a partially ordered set, then we write $x \parallel y$ if $x \nleqslant y$ and $y \nleqslant x$. We may write $x \geqslant y$ whenever $y \leqslant x$. We may also write $x < y$ whenever $x \leqslant y$ and $x \neq y$. Finally, we will write $[n]$ to denote $\{1, \dots, n\}$.

Definition 1. *A* workflow specification *is a partially ordered set of steps $(S, \leqslant)$. An* authorization policy *for a workflow specification is a relation $A \subseteq S \times U$. A* workflow authorization schema *is a tuple $(S, U, \leqslant, A)$, where $(S, \leqslant)$ is a workflow specification and A is an authorization policy.*

If $s < s'$ then s must be performed before s' in any instance of the workflow; if $s \parallel s'$ then s and s' may be performed in either order. User u is authorized to perform step s only if $(s, u) \in A$.[2] We assume that for every step $s \in S$ there exists some user $u \in U$ such that $(s, u) \in A$.

Definition 2. *Let $(S, U, \leqslant, A)$ be a workflow authorization schema. A* plan *is a function $\pi : S \to U$. A plan π is* authorized *for $(S, U, \leqslant, A)$ if $(s, \pi(s)) \in A$ for all $s \in S$.*

Definition 3. *Let $S_1, S_2 \subseteq S$, where either S_1 or S_2 (or both) is a singleton set and $\rho \in Rel(U)$. Then a* workflow constraint *has the form (ρ, S_1, S_2). A* constrained workflow authorization schema *is a tuple $(S, U, \leqslant, A, C)$, where C is a set of workflow constraints.*

Definition 4. *A plan $\pi : S \to U$* satisfies *constraint (ρ, S_1, S_2) if there exist $s_1 \in S_1$ and $s_2 \in S_2$ such that $(\pi(s_1), \pi(s_2)) \in \rho$. Given a constrained workflow authorization schema $(S, U, \leqslant, A, C)$, a plan π is* valid *if it is authorized and it satisfies all constraints in C.*

The above definition of a workflow constraint was introduced by Wang and Li [17]. It generalizes that of Crampton [5], in which both S_1 and S_2 were singleton sets. A further natural generalization is to define a constraint to be a triple (ρ, S_1, S_2), where S_1 and S_2 are arbitrary subsets of S, and to say that plan π satisfies constraint (ρ, S_1, S_2) provided there exist $s_i \in S_i$ such that $(\pi(s_1), \pi(s_2)) \in \rho$. In order to facilitate direct comparison with the work of Wang and Li, we defer the discussion of these more general constraints until Section 5.1. We will write s, rather than the mathematically correct $\{s\}$, when $s \in S$ appears as a singleton in a constraint. When ρ is a symmetric relation (meaning that (ρ, s, S') is satisfied if and only if (ρ, S', s) is satisfied), we will write constraints in the form (ρ, s, S'); that is, the set of cardinality greater than or equal to 1 appears after the set of cardinality 1.

The plan π satisfies constraint $(\rho_=, s, s')$, for example, if the same user is assigned to both steps by π, and satisfies constraint $(\rho_{\neq}, s, s')$ if different users are assigned to s and s'. In other words, these represent, respectively, binding-of-duty and separation-of-duty constraints. Abusing notation in the interests of readability, we will replace $\rho_=$ and $\rho_{\neq}$ by $=$ and $\neq$, respectively. The constraint $(\neq, s, S')$, for example, is satisfied by plan π if there is some step in S' to which some user other than the one who performed step s is assigned.

We may now define the workflow satisfiability problem, as defined by Wang and Li [17].

Workflow Satisfiability Problem (WSP)
Input: A constrained workflow authorization schema $(S, U, \leqslant, A, C)$
Output: A valid plan $\pi : S \to U$ or an answer that there exists no valid plan

Note that the above definition does not make any reference to the ordering on the set of steps. The original definition, as formulated by Crampton [5], required an algorithm for WSP to output an execution schedule (a topological sort of $(S, \leqslant)$) in addition to a valid plan. Again, to facilitate direct comparison with the work of Wang and Li on the fixed parameter tractability of WSP, we defer discussion of Crampton's version of the problem until Section 5. Henceforth, we will write WSP$(\rho_1, \dots, \rho_t)$ to denote a special case of WSP in which all constraints have the form (ρ_i, s, S') or (ρ_i, S', s) for some $i \in [t]$ and for some $s \in S$ and $S' \subseteq S$.

3. ORGANIZATIONAL CONSTRAINTS

The work of Crampton [5, § 2] and of Wang and Li [17, Examples 1, 2] has noted that a constraint of practical interest is that users performing two steps must be from the same department.[3] Indeed, in our example illustrated in Figure 1 one could require that the two users who perform steps s_3 and s_5 belong to the same department. Note, however, that we will still require that these two users must be different. Similarly, we might wish to insist that the user who approves the purchase order (step s_2) belongs to the same department as the user who creates the order (step s_1).

In short, one can imagine many practical situations in which some auxiliary information defines an equivalence relation on the set of users (membership of department, for example) and that we may wish to require that two steps are performed by users belonging to either the same department or to different departments. In this section, we introduce two relations that allow us to model organizational structures, in which users are partitioned (possibly at several levels) into different organizational units, such as departments.

It is well known that an equivalence relation $\sim$ on a set X induces a partition of X into equivalence classes. Conversely, a partition of X gives rise to an equivalence relation. Given an equivalence relation on U, we write $\sim$ in a constraint to indicate that two users must belong to the same equivalence class and $\not\sim$ to denote that two users must belong to different equivalence classes. Hence, $(\sim, s_3, s_5)$ would indicate the requirement that the signing and countersigning of the goods received note must be performed by users belonging to the same equivalence class (department, in this example).

[2] In practice, the set of authorized step-user pairs, A, will not be defined explicitly. Instead, A will be inferred from other access control data structures. In particular, $\mathsf{R}^2\mathsf{BAC}$ – the role-and-relation-based access control model of Wang and Li [17] – introduces a set of roles R, a user-role relation $UR \subseteq U \times R$ and a role-step relation $SA \subseteq R \times S$ from which it is possible to derive the steps for which users are authorized. For all common access control policies (including $\mathsf{R}^2\mathsf{BAC}$), it is straightforward to derive A. We prefer to use A in order to simplify the exposition.

[3] However, little is known about the complexity of the WSP when such constraints are used, a deficiency we address in the next section.

3.1 Organizational Hierarchies

We now show how we can use multiple partitions to define an organizational hierarchy. In Section 4.4, we describe a fixed-parameter tractable algorithm to solve WSP in the presence of constraints defined over such structures.

Let S be a set. An *n-partition* of S is an n-tuple $(F_1, \dots, F_n)$ such that $F_1 \cup \dots \cup F_n = S$ and $F_i \cap F_j = \emptyset$ for all $i \neq j \in [n]$. We will refer to the elements of an n-partition as *blocks*.[4]

Definition 5. *Let $(X_1, \dots, X_p)$ and $(Y_1, \dots Y_q)$ be p- and q-partitions of the same set. We say that $(Y_1, \dots Y_q)$ is a* refinement *of $(X_1, \dots, X_p)$ if for each $i \in [q]$ there exists $j \in [p]$ such that $Y_i \subseteq X_j$.*

Definition 6. *Let U be the set of users in an organization. An* organizational ℓ-hierarchy *is a collection of ℓ partitions of U, $U^{(1)}, \dots, U^{(\ell)}$, where $U^{(i)}$ is a refinement of $U^{(i+1)}$.*

The ith partition is said to be the ith *level* of the hierarchy. Each member of $U^{(i)}$ is a subset of U; we write $u^{(i)}$ to denote a block in the ith level of the hierarchy.

A constraint of the form $(\sim_i, s_1, s_2)$, for example, is satisfied by plan π if $\pi(s_1), \pi(s_2) \in u^{(i)}$ for some $u^{(i)} \in U^{(i)}$. Note, however, that we may still define a constraint $(\neq, s_1, s_2)$ which requires that the steps s_1 and s_2 are performed by different users.

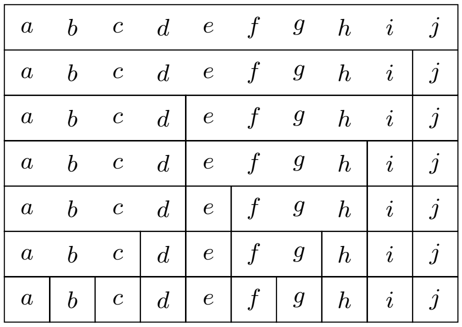
More generally, a constraint of the form $(\sim_i, S_1, S_2)$ is satisfied by plan π if there exists $s_1 \in S_1$ and $s_2 \in S_2$ such that $\pi(s_1)$ and $\pi(s_2)$ belong to the same block in $U^{(i)}$. A constraint of the form $(\not\sim_i, S_1, S_2)$ is satisfied by π if there exist $s_1 \in S_1$ and $s_2 \in S_2$ such that $\pi(s_1)$ and $\pi(s_2)$ belong to different blocks in $U^{(i)}$. Note that if π satisfies $(\sim_i, S_1, S_2)$, then it satisfies $(\sim_j, S_1, S_2)$ for all $j > i$. Conversely, if π satisfies $(\not\sim_i, S_1, S_2)$, then it also satisfies $(\not\sim_j, S_1, S_2)$ for all $j < i$. In other words, for each $S_1, S_2 \subseteq S$, we may and will assume without loss of generality that there is at most one constraint of the form $(\sim_i, S_1, S_2)$ and at most one constraint of the form $(\not\sim_j, S_1, S_2)$.

We now introduce the notion of a canonical hierarchy. Informally, each level of a canonical hierarchy is different, the top level comprises a single block and the bottom level comprises the set of all singleton blocks. A canonical hierarchy is shown in Figure 2, in which $a, \dots, j$ represent users and the rectangles define the partition blocks. Note that each level is a refinement of the one above.

More formally, we have the following definition.

Definition 7. *Let $\mathcal{H} = U^{(1)}, \dots, U^{(\ell)}$, where $U^{(i)}$ is a refinement of $U^{(i+1)}$, be a hierarchy. We say $\mathcal{H}$ is* canonical *if it satisfies the following conditions: (i) $U^{(i)} \neq U^{(i+1)}$; (ii) $U^{(\ell)}$ is a 1-partition containing the set U; (iii) $U^{(1)}$ is an n-partition containing every singleton set (from U).*

Let $U^{(1)}, \dots, U^{(\ell)}$ be some hierarchy and let C be a set of workflow constraints. We conclude this section by showing how we may convert the hierarchy into a canonical hierarchy by first removing duplicate levels, adding suitable top and bottom levels (if required), and making appropriate adjustments to C. More formally, we perform the following operations:



a	b	c	d	e	f	g	h	i	j
a	b	c	d	e	f	g	h	i	j
a	b	c	d	e	f	g	h	i	j
a	b	c	d	e	f	g	h	i	j
a	b	c	d	e	f	g	h	i	j
a	b	c	d	e	f	g	h	i	j
a	b	c	d	e	f	g	h	i	j

Figure 2: A canonical organizational hierarchy

- If $U^{(i)} = U^{(i+1)}$ for some i then we replace all constraints of the form $(\sim_{i+1}, S_1, S_2)$ and $(\not\sim_{i+1}, S_1, S_2)$ with constraints of the form $(\sim_i, S_1, S_2)$ and $(\not\sim_i, S_1, S_2)$, respectively. We then remove $U^{(i+1)}$ from the hierarchy as there are now no constraints that apply to $U^{(i+1)}$.
- If no partition in the hierarchy has one element (consisting of a single block U), then add such a partition to the hierarchy. Clearly every partition is a refinement of the 1-partition (U).
- If no partition in the hierarchy has n elements, then add such a partition to the hierarchy. Clearly such a partition is a refinement of every other partition.
- Finally, we renumber the levels and the constraints where appropriate with consecutive integers.

The conversion of a hierarchy to canonical form can be performed in $O(\ell n + c)$ time (since we require $O(\ell n)$ time to find all layers that may be deleted and then delete them, and $O(c)$ time to update the constraints). The number of levels in the resulting canonical hierarchy is no greater than $\ell + 2$.

3.2 Organizational Hierarchies from Management Structures

We now illustrate how organization hierarchies may be constructed in a systematic fashion from management structures. Given a set of users U, we assume that an organization defines a hierarchical binary relation $\succcurlyeq$ on U in order to specify management responsibilities and reporting lines. We assume that the Hasse diagram of $(U, \succcurlyeq)$ is a directed tree in which non-leaf nodes represent users with some managerial responsibility and edges are directed from root node to leaf nodes. Let $G_{\text{man}} = (U, E_{\text{man}})$ denote the Hasse diagram of $(U, \succcurlyeq)$. The fact that G_{man} is a tree means that no user has more than one manager. A user u has direct responsibility for (or is the line manager of) user v if $(u, v) \in E_{\text{man}}$. We also assume that the out-degree of a non-leaf node is at least two. Both of these assumptions would seem to be reasonable for most organizations with a hierarchical management structure. (The study of more exotic management structures such as matrix management is deferred to future work.)

We now describe one method by which an organizational hierarchy may be derived from a management tree. Given a management tree G_{man} we iteratively construct management trees with fewer and fewer nodes as follows:

1. we first identify every sub-tree in which there is a single non-leaf node;

[4] One or more blocks in an n-partition may be the empty set.

2. for each such sub-tree we form a single leaf node whose label is formed from the labels for the respective leaf nodes;
3. for each resulting sub-tree we form a single node whose label is formed from the labels of the child and parent nodes.

We then repeat for the resulting tree, terminating when we have a tree containing a single node. The above procedure is illustrated in Figure 3. The figure shows a sequence of trees, the first of which defines the management tree in which each node is labeled with a single user.

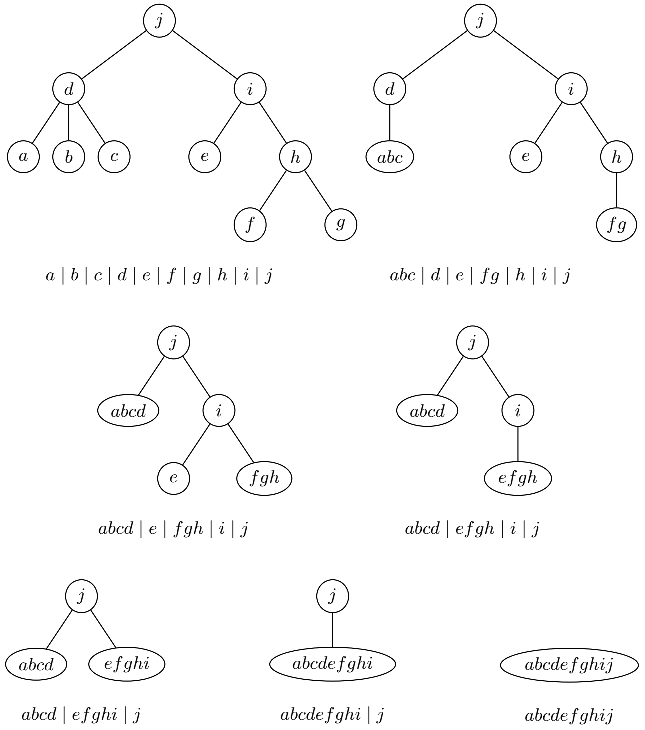

j
d
i
a
b
c
e
h
f
g
$a \mid b \mid c \mid d \mid e \mid f \mid g \mid h \mid i \mid j$
j
d
i
abc
e
h
fg
$abc \mid d \mid e \mid fg \mid h \mid i \mid j$
j
abcd
i
e
fgh
$abcd \mid e \mid fgh \mid i \mid j$
j
abcd
i
efgh
$abcd \mid efgh \mid i \mid j$
j
abcd
efghi
$abcd \mid efghi \mid j$
j
abcdefghi
$abcdefghi \mid j$
abcdefghij
$abcdefghij$

Figure 3: Building the blocks of an organizational hierarchy from a management tree

Each management tree thus derived is associated with a partition; the corresponding partition of U is written below each tree in Figure 3, with a vertical bar indicating the block boundaries. By construction, the collection of partitions forms a canonical organizational hierarchy. The organizational hierarchy derived from the management tree in Figure 3 is displayed in Figure 2. Note that the number of levels in the organizational hierarchy is equal to $2p + 1$, where p is the number of edges in the longest directed path in G_{man}.

Having constructed the organizational hierarchy, we may now define constraints on step execution. We will use our purchase order workflow from Figure 1 as an example and the organizational hierarchy in Figure 2.

We could, for example, define the constraint $(\sim_5, s_1, s_2)$. In the absence of other constraints, this constraint means that users from the set $\{a, b, c, d\}$ or $\{e, f, g, h, i\}$ (which we might suppose represent two distinct departments within the management structure) or user j could raise (step s_1) and approve (step s_2) purchase orders, but an attempt by a user from one department to approve an order raised by a member of another department would violate the constraint.

We could define a second constraint $(\nsim_4, s_1, s_2)$, which means that user i must perform one of s_1 and s_2 (and also means that no user from $\{a, b, c, d, j\}$ can perform either s_1 or s_2 because there would no way to simultaneously satisfy constraints $(\sim_5, s_1, s_2)$ and $(\nsim_4, s_1, s_2)$). If we assume that junior members of the department (users e, f, g and h) are not authorized to approve purchase orders, the collective effect of the two constraints above and the authorization policy is to require that (a) purchase orders are only approved by managers, and (b) purchase orders are only raised by junior members of staff.

Pursuing the last point briefly, it has long been recognized that a limitation of role-based access control is the "feature" that (senior) users assigned to the most powerful roles accrue all the permissions of more junior roles (see [14], for example). It is interesting to note that the constraints and the method of constructing an organizational hierarchy described above can be used to restrict the steps that senior managers can perform.

In summary, we believe that our definition of organizational hierarchy provides an appropriate way of modeling hierarchical management structures and supports the specification of constraints that provide greater flexibility than those in the literature [2, 5, 17]. Moreover, as we will see in the next section, the complexity of WSP for these new constraints remains fixed-parameter tractable.

Finally, we note that there are several ways in which the construction of an organizational hierarchy from a management tree described above could be modified: for example, each iteration might collapse a single sub-tree rather than all sub-trees or each iteration might immediately collapse the root node and all the leaf nodes into a single node. Each construction will give rise to different organizational hierarchies, some with more levels, some with fewer. These organizational hierarchies each admit the specification of different constraints. The study of such hierarchies and the utility of the constraints that can be defined over them will be the subject of future work.

4. FPT ALGORITHMS FOR WSP

In this section, we prove a new bound for the fixed parameterized complexity of WSP$(\neq, =)$. We also prove that WSP remains fixed-parameter tractable when we include constraints based on equivalence relations and organizational hierarchies. In order to make the paper self-contained, we first provide a short overview of fixed-parameter complexity and state an important, recent result in this area.

4.1 Parameterized Complexity

A naïve approach to solving WSP would consider every possible assignment of users to steps in the workflow. There are n^k such assignments if there are n users and k steps, so an algorithm of this form would have complexity $O(cn^k)$, where c is the number of constraints. Moreover, Wang and Li showed that WSP is NP-hard, by reducing Graph k-Colorability to WSP$(\neq)$ [17, Lemma 3]. In short, WSP is hard to solve in general. The importance of finding an efficient algorithm for solving WSP [5] led Wang and Li to look at the problem from the perspective of parameterized complexity [17].

Suppose we have an algorithm that solves an NP-hard problem in time $O(f(k)n^d)$, where n denotes the size of the input to the problem, k is some (small) parameter of the problem, f is some function in k only, and d is some constant (independent of k and n). Then we say the algorithm is a *fixed-parameter tractable* (FPT) algorithm. If a problem can be solved using an FPT algorithm then we say that it is an *FPT problem* and that it belongs to the class FPT. Wang and Li showed that WSP$(=, \neq)$ is FPT: specifically, they constructed an algorithm that runs in time $O(k^{k+1}(k-1)^{k2^{k-1}}n)$, where k is the number of steps in the workflow and n is the size of the input to the problem [17, Theorem 8]; it follows that WSP$(=, \neq)$ is FPT.

Wang and Li also proved that WSP is W[1]-hard in general [17, Theorem 10]. By definition, FPT is a subset of W[1] and a parameterized analog of Cook's Theorem [8] as well as the Exponential Time Hypothesis [9,11] strongly support the widely held view that FPT is not equal to W[1]. One of the main contributions of this paper is to extend the set of special cases of WSP that are known to be FPT.

When the runtime $O(f(k)n^d)$ is replaced by the much more powerful $O(n^{f(k)})$, we obtain the class XP, where each problem is polynomial-time solvable for any fixed value of k: There is an infinite number of parameterized complexity classes between FPT and XP (for each integer $t \geq 1$, there is a class W[t]) and they form the following tower: FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ W[P] $\subseteq$ XP. For formal definitions of classes W[t], see [8, 9], for example.

Henceforth, we often write $\widetilde{O}(T)$ instead of $O(T \log^d T)$ for any constant d. That is, we use the notation $\widetilde{O}$ to suppress polylogarithmic factors. This notation is often used in the literature on algorithms – see, for example, [3,12] – to avoid cumbersome runtime bounds.

Our work on WSP makes use of the following result, due to Björklund *et al.* [3], regarding the complexity of the Max Weighted Partition problem.

Max Weighted Partition
Input: A set S of k elements and n functions ϕ_i, $i \in [n]$, from 2^S to integers from the range $[-M, M]$ $(M \geq 1)$.
Output: An n-partition $(F_1, \ldots, F_n)$ of S that maximizes $\sum_{i=1}^{n} \phi_i(F_i)$.

Theorem 1. Max Weighted Partition *can be solved in time* $\widetilde{O}(2^k n^2 M)$.

4.2 A New Algorithm for WSP

We now introduce a new method for solving WSP. The basic idea is to construct a valid plan by partitioning the set of steps S into blocks of steps, each of which is allocated to a single (authorized) user.

More formally, let π be a valid plan for a workflow $(S, U, \leqslant, A, C)$ and define an equivalence relation $\sim_\pi$ on S, where $s \sim_\pi s'$ iff $\pi(s) = \pi(s')$. We denote the set of equivalence classes of $\sim_\pi$ by S/π and write $[s]_\pi$ to denote the equivalence class containing s.

Now it is easy to see that there are certain subsets S' of S for which there cannot exist a valid plan π such that $S' \in S/\pi$. In particular, consider the constraint $(\neq, s, s')$, for example. Then, for any valid plan π, it must be the case that $[s]_\pi \neq [s']_\pi$. In other words, there does not exist a valid plan π such that $\{s, s'\} \in S/\pi$. We call such sets *ineligible*.

We solve WSP by computing a partition of S in which each block of the partition is assigned to a different user and each user is authorized to perform all steps in the block to which she is assigned. Our strategy for solving WSP, then, is the following:

1. take the powerset of S and eliminate all subsets that are ineligible;
2. for each remaining set F, identify which users are authorized for all steps in F;
3. construct a partition of S such that each block is allocated to a user authorized for each step in that block and no user is allocated to more than one block (we can find such a partition using Theorem 1).

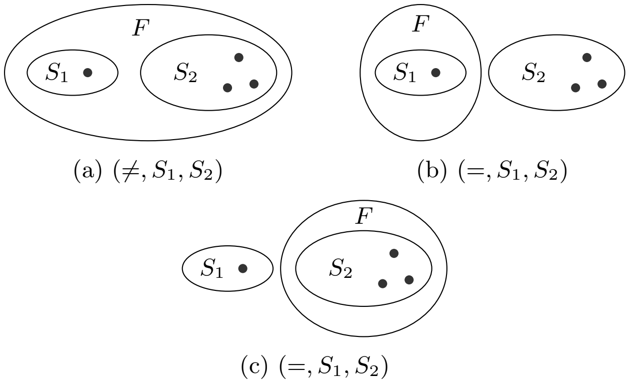
The resulting partition and allocation of users to blocks in that partition defines a valid plan.

Remark 1. *Note that a constraint* $(\neq, S_1, S_2)$ *is equivalent*[5] *to* $(\neq, S_1, S_2 \setminus S_1)$ *and that a constraint* $(=, S_1, S_2)$ *is always satisfied when* $S_1 \cap S_2 \neq \emptyset$. *Thus, in the rest of the paper we assume that, in constraints* $(\neq, S_1, S_2)$ *and* $(=, S_1, S_2)$*, the sets* S_1 *and* S_2 *are disjoint.*

It is easy to determine whether a set is ineligible when the set of constraints only contains constraints of the form $(\neq, S_1, S_2)$ or of the form $(=, S_1, S_2)$, when S_1 is a singleton. In particular, we have the following simple result, illustrated schematically in Figure 4.

Proposition 1. *Let* $(S, U, \leqslant, A, C)$ *be a workflow and let* C *contain constraints* $(\neq, S_1, S_2)$ *and* $(=, S'_1, S'_2)$*, where* S_1 *and* S'_1 *are singleton sets. Then* $F \subseteq S$ *is ineligible if either of the following conditions hold: (i)* $F \supseteq S_1 \cup S_2$*; (ii)* $F \cap (S'_1 \cup S'_2) = S'_i$*, for some* $i \in \{1, 2\}$.

Proof. Suppose (in order to obtain a contradiction) that $F \supseteq S_1 \cup S_2$ is eligible. Then, by definition, there must exist a valid plan π such that $F \in S/\pi$, which, in turn, means that a single user has been allocated to all steps in F by π. But such an allocation violates constraint $(\neq, S_1, S_2)$ and hence the plan is not valid. A similar proof by contradiction can be used to establish the second result. □



(a) $(\neq, S_1, S_2)$

(b) $(=, S_1, S_2)$

(c) $(=, S_1, S_2)$

Figure 4: Ineligible sets F for constraints (ρ, S_1, S_2), $\rho \in \{=, \neq\}$

[5] Equivalent in the sense that a plan π satisfies the former constraint iff it satisfies the latter.

Example 1. *Using the workflow in Figure 1, for example, we find that any subset of cardinality four or greater is ineligible; and all subsets of cardinality three are ineligible, except for $\{s_1, s_3, s_6\}$, $\{s_2, s_4, s_5\}$ and $\{s_2, s_5, s_6\}$. We also have that $\{s_1\}$ and $\{s_3\}$ are ineligible (because of the binding-of-duty constraint on s_1 and s_3). Similarly any superset of $\{s_1\}$ that does not contain s_3 is ineligible and vice versa.*

If we were to find that there existed a user u, authorized for s_1, s_3 and s_6, and a different *user v, authorized for s_2, s_4 and s_5, then we would know that this instance of* WSP *is satisfiable. Note the requirement that u and v be different: if they were the same then u would perform all steps, violating all separation-of-duty constraints. Conversely, if we were to find that there is no user authorized for s_1 and s_3 then we would know that the workflow is unsatisfiable.*

Suppose now that we replace the constraint $(=, s_1, s_3)$ with the constraint $(=, s_1, \{s_3, s_5\})$, which has the effect of requiring that the same user that creates the order either signs or countersigns the goods received note (but not both, given the constraint $(\neq, s_3, s_5)$). Then $\{s_1\}$ is ineligible (as before) but $\{s_3\}$ and $\{s_1, s_5\}$ are no longer ineligible; any superset of $\{s_3, s_5\}$ is ineligible.

Theorem 2. WSP$(=,\neq)$ *can be solved in time $\widetilde{O}(2^k(c + n^2))$, where k is the number of steps, c is the number of constraints and n is the number of users.*

Proof. Consider an instance $\mathcal{W} = (S, U, \leqslant, A, C)$ of WSP$(=,\neq)$. We construct a binary matrix with n rows (indexed by elements of U) and 2^k columns (indexed by elements of 2^S) in which the entry indexed by $u \in U$ and $F \subseteq S$ is defined to be 0 if and only if $F \neq \emptyset$ is ineligible or there exists $s \in F$ such that $(s, u) \notin A$. (Note that every entry in the column labeled with $\emptyset$ is set to 1.) In other words, each non-zero matrix entry represents a set of steps that could be assigned to a single user in a valid plan.

The matrix defined above encodes a family of functions $\{\phi_u\}_{u \in U}$, $\phi_u : 2^S \to \{0, 1\}$. We now solve Max Weighted Partition on input S and $\{\phi_u\}_{u \in U}$. Observe that $\sum_{i=1}^{n} \phi_i(F_i) \leqslant n$, with equality if and only if $\mathcal{W}$ is satisfiable.

We now consider the complexity of the above algorithm. We consider each of the $2^k n$ elements in the matrix. We set the entry for (u, F) equal to 0 if one of the following conditions holds:

- there exists a constraint of the form $(\neq, S_1, S_2)$ with $S_1 \cup S_2 \subseteq F$;
- there exists a constraint of the form $(=, S_1, S_2)$ with $F \cap (S_1 \cup S_2) = S_i$ for $i \in \{1, 2\}$;
- there exists $s \in F$ such that $(s, u) \notin A$.

The first two conditions check whether F is ineligible (see Proposition 1), while the last condition tests whether u is authorized for all the steps in F. We can identify the ineligible sets in time $O(2^k ck) = \widetilde{O}(2^k c)$. We can then construct the matrix in $O(2^k nk) = \widetilde{O}(2^k n)$ time. Finally, we can solve Max Weighted Partition in $\widetilde{O}(2^k n^2)$ time. Thus, the total time required to solve WSP$(=,\neq)$ is in $\widetilde{O}(2^k(c+n^2))$. □

The special case of the workflow satisfiability problem WSP$(\neq)$ was studied by Wang and Li from the perspective of fixed-parameter tractability; the complexity of their algorithm is $O(k^{k+1}N) = 2^{O(k \log k)}N$, where N is the size of the input [17, Lemma 8]. Fellows *et al.* considered the fixed-parameter tractability of a special case of the *constraint satisfaction problem* [15] in which all constraints have the same form. With these restrictions, the constraint satisfaction problem is identical to WSP in which all constraints have the form $(\neq, s_1, s_2)$. The algorithm of Fellows *et al.* has complexity $O(k!kn) = 2^{O(k \log k)}n$, where n is the number of users [10, Theorem 3.1]. Theorem 2 above exhibits an algorithm with complexity $\widetilde{O}(2^k(c + n^2))$, so we have improved the exponential factor of the best known bounds for WSP$(\neq)$ in terms of the function in k (which is often the dominating factor in the complexity of FPT algorithms). We now demonstrate that it is impossible, assuming the well-known *Exponential Time Hypothesis* [11], to improve this result to any significant degree.

Exponential Time Hypothesis
There exists a real number $\epsilon > 0$ such that 3-SAT cannot be solved in time $O(2^{\epsilon n})$, where n is the number of variables.

Theorem 3. *Even if there are just two users,* WSP$(\neq)$ *cannot be solved in time $\widetilde{O}(2^{\epsilon k})$ for some positive real ϵ, where k is the number of steps, unless the Exponential Time Hypothesis fails.*

The proof of this result (and Theorem 5) can be found in the appendix.

4.3 Equivalence Relation Constraints

Observe that the equality relation is an equivalence relation in which each of the equivalence classes comprises a single element, and so $=$ and $\neq$ may be modelled as instances of $\sim$ and $\not\sim$. Henceforth, therefore, we only consider constraints of the form $(\sim, S_1, S_2)$ and $(\not\sim, S_1, S_2)$.

In Proposition 2 below, we prove that WSP$(\sim, \not\sim)$ is FPT, thereby extending the class of workflow specifications for which the satisfiability problem is in FPT. In turn, Proposition 2 will be significantly extended in Theorem 4.

Proposition 2. *For any user set U and any equivalence relation $\sim$ defined on U,* WSP$(\sim, \not\sim)$ *is FPT.*

Proof. Consider an instance of the problem $\mathcal{W} = (S, U, \leqslant, A, C)$ and let $V_1, \ldots, V_m$ be the equivalence classes of $\sim$. Then consider the following workflow specification: $\mathcal{W}' = (S, U', \leqslant, A', C')$, where

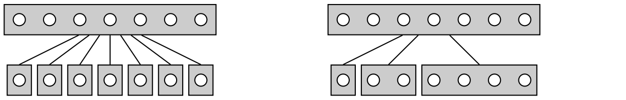
- $U' = \{V_1, \ldots, V_m\}$;
- $A' \subseteq S \times U'$ and $(s, V_i) \in A'$ if there exists $u \in V_i$ such that $(s, u) \in A$;
- each constraint of the form $(\sim, S_1, S_2)$ in C is replaced by $(=, S_1, S_2)$ in C'; and
- each constraint of the form $(\not\sim, S_1, S_2)$ in C is replaced by $(\neq, S_1, S_2)$ in C'.

It is easy to see that $\mathcal{W}$ is satisfiable if and only if $\mathcal{W}'$ is, and deciding the satisfiability of $\mathcal{W}'$ is FPT by Theorem 2. □

4.4 Organizational Hierarchy Constraints

We have seen that if we are given a single equivalence relation and only use the binary relations $\sim$ and $\not\sim$ then WSP$(\sim, \not\sim)$ is equivalent to WSP$(=,\neq)$, which is known to be FPT. Moreover, we prove in Theorem 4 that the problem remains in FPT for such hierarchies. In fact, the results in Theorem 2 and Proposition 2 correspond to special cases of

Theorem 4, in which the hierarchy has two levels. Figure 5 illustrates these hierarchies, where each user is represented by an unfilled circle and blocks of users are enclosed by a rectangle. Conversely, it is these special cases that provide the foundation for the bottom-up iterative method that we use in the proof of Theorem 4 to solve WSP for more complex hierarchical structures.



(a) Single users (b) Non-trivial equivalence relation

Figure 5: Two-level hierarchies

THEOREM 4. *Given a workflow* $(S, U, \leqslant, A, C)$ *and a canonical hierarchy with* ℓ *levels,* $\mathrm{WSP}(\sim_1, \not\sim_1, \ldots, \sim_\ell, \not\sim_\ell)$ *can be solved in time* $\widetilde{O}(cn2^k + n^2 3^k)$*, where* n*,* k *and* c *are the numbers of users, steps and constraints, respectively.*

To prove the above result, we iteratively identify ineligible sets, from the bottom of the hierarchy up, for particular types of blocks in the hierarchy (those shaded in Figure 6) and solve multiple instances of WSP for each of those blocks. Theorem 2 is, essentially, a special case of the above result, in which the canonical hierarchy contains two levels, where $U^{(1)} = (\{u_1\}, \ldots, \{u_n\})$ and $U^{(2)} = (U)$.

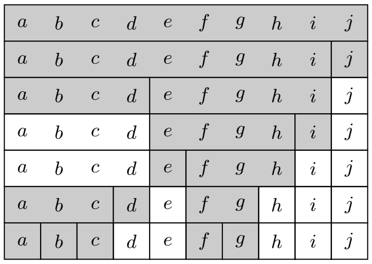


Figure 6: The canonical hierarchy of Figure 3 with its significant blocks shaded

PROOF OF THEOREM 4. Each level in a canonical hierarchy is a refinement of the one below it and no two levels are equal, so we have $n = |U^{(1)}| > \cdots > |U^{(\ell)}| = 1$, and we may conclude that $\ell \leqslant n$.

We say $V \in U^{(i)}$ is *significant* if $V \notin U^{(i+1)}$. The significant blocks are shaded in the hierarchy shown in Figure 6. We define the *level range* of V to be an interval $[a, b]$, where a is the least value i such that $V \in U^{(i)}$ and b is the largest value i such that $V \in U^{(i)}$. The level range of block $\{a, b, c, d\}$ in Figure 6 is $[3, 5]$, for example.

Each significant block V with level range $[a, b]$, $a > 1$, can be partitioned into blocks in level $(a - 1)$. (Block $\{a, b, c, d\}$ is partitioned into $\{\{a, b, c\}, \{d\}\}$ in level 2, for example.) We denote this set of blocks by $\Delta(V)$. It is easy to see that the graph $G = (\mathcal{V}, E)$, where $\mathcal{V}$ is the set of significant blocks and $(V_1, V_2) \in E$ if $V_1 \in \Delta(V_2)$, is a tree.

For each significant block V in the hierarchy with level range $[a, b]$, working from the bottom level to the top level of the hierarchy, we compute a function $\phi_V : 2^S \to \{0, 1\}$, where $\phi_V(F) = 1$ if and only if there exists an assignment of authorized users from V to steps in F that satisfies all the constraints at level b and below. Informally, we will be solving multiple instances of WSP for the sub-tree rooted at V in order to compute ϕ_V.

Each significant block V with level range $[1, b]$ comprises a single user (see Figure 6). For $F \subseteq S$ we set $\phi_V(F) = 0$ if one of the following conditions hold:

- there exists a constraint of the form $(\not\sim_t, S_1, S_2)$, $t \in [1, b]$, such that $S_1 \cup S_2 \subseteq F$;
- there exists a constraint of the form $(\sim_t, S_1, S_2)$, $t \in [1, b]$, such that $F \cap (S_1 \cup S_2) = S_i$ for $i \in \{1, 2\}$;
- there exists $s \in F$ such that $(s, u) \notin A$.

For each significant block V, we now compute a binary matrix with rows indexed by members of $\Delta(V)$ and columns indexed by subsets of S. The entry indexed by $W \in \Delta(V)$ and $F \subseteq S$ is defined to be 0 if and only if one of the following conditions holds:

- there exists a constraint of the form $(\not\sim_t, S_1, S_2)$, $t \in [a, b]$, such that $S_1 \cup S_2 \subseteq F$;
- there exists a constraint of the form $(\sim_t, S_1, S_2)$, $t \in [a, b]$, such that $F \cap (S_1 \cup S_2) = S_i$ for $i \in \{1, 2\}$;
- $\phi_W(F) = 0$.

A non-zero matrix entry indicates that the steps in F could be assigned to the block W (which implies that no constraints in levels $1, \ldots, a - 1$ would be violated) and that no constraints would be violated in levels $a, \ldots, b$ by allocating a single block to F. The matrix encodes a family of functions $\{\phi_W\}_{W \in \Delta(V)}$, $\phi_W : 2^S \to \{0, 1\}$. We then solve MAX WEIGHTED PARTITION with inputs $F \subseteq S$, and $\{\phi_W\}$ and set $\phi_V(F) = 1$ if MAX WEIGHTED PARTITION returns a partition such that $\sum_W \phi_W(F) = |\Delta(V)|$. Note that U is a significant block (being the single block in the top level of the hierarchy). Note also that WSP is satisfiable if and only if $\phi_U(S) = 1$.

This completes the description of the algorithm. We now consider its complexity. Consider the significant block V with level range $[a, b]$, $a > 1$, and m children. (It takes time $O(\ell) = O(n)$ to compute the level range of V.) Note that we can initialize the matrix using the functions $\{\phi_W\}$ that were computed for the $(a - 1)$th level. We then change the non-zero entries for those $F \subseteq S$ that are ineligible. As in the proof of Theorem 2, the identification of the ineligible sets takes time $O(2^k ck) = \widetilde{O}(2^k c)$, so the time taken to update the matrix is $\widetilde{O}(2^k cm)$.

The time taken to compute ϕ_V for V with level range $[1, b]$ is dominated by the time taken to compute the ineligible sets, which takes $\widetilde{O}(2^k c)$. For V with level range $[a, b]$, $a > 1$, we solve MAX WEIGHTED PARTITION for all $F \subseteq S$. The time taken to solve a single instance of MAX WEIGHTED PARTITION is $\widetilde{O}(2^{|F|} m^2)$. Hence, the time taken to solve all instances of WSP (for V) is $\widetilde{O}(3^k m^2)$.[6] Finally, the total time taken to compute ϕ_V is $\widetilde{O}(n + cm2^k + m^2 3^k)$.

We next count the number of significant blocks. As we observed earlier, the set of significant blocks ordered by subset inclusion forms a tree. Moreover, every non-leaf node in G has at least two children, which implies that G has no more than $2n - 1$ nodes (so $|\mathcal{V}| \leqslant 2n - 1$).

Finally, let m_V denote the number of children of a signif-

[6] Note that $\sum_{S' \subseteq S} 2^{|S'|} = \sum_{i=0}^{k} \binom{k}{i} 2^i = 3^k$.

icant block V. Then

$$\sum_{V \in \mathcal{V}} \widetilde{O}(m_V) = \sum_{V \in \mathcal{V}} O(m_V \log^b m_V)$$

for some $b \geqslant 0$. But

$$\sum_{V \in \mathcal{V}} m_V \log^b m_V \leqslant \max_{V \in \mathcal{V}} \log^b m_V \sum_{V \in \mathcal{V}} m_V = O(n \log^b n) = \widetilde{O}(n).$$

Hence $\sum_{V \in \mathcal{V}} \widetilde{O}(m_V) = \widetilde{O}(n)$. Similarly, we can show that $\sum_{V \in \mathcal{V}} \widetilde{O}(m_V^2) = \widetilde{O}(n^2)$. Hence, we conclude that the total time taken to compute ϕ_V for all V is $\widetilde{O}(cn2^k + n^2 3^k)$. $\square$

5. EXTENSIONS TO WSP

The workflow satisfiability problem studied in this paper was defined by Wang and Li, which, in turn, was based on the workflow model proposed by Crampton [5]. Wang and Li's formulation of the problem differs in two crucial aspects from that of Crampton. Both aspects are related to the definition of constraints. On the one hand, Wang and Li make the constituent sets of the constraints more complex: where Crampton assumed that both sets were singletons, Wang and Li allow at most one of those sets to have cardinality greater than one; this means that more complex relationships between the users that perform the steps can be specified using constraints. On the other hand, Wang and Li ignore the partial ordering on workflow steps when defining constraints, whereas Crampton made a distinction between the constraints (ρ, s_1, s_2) and (ρ, s_2, s_1) and disallowed a constraint of the form (ρ, s_2, s_1) if $s_1 < s_2$.

In this section, we discuss several different ways in which WSP may be extended. First, we extend the definition of constraints so that they are more general than those considered by Wang and Li. Second, we consider the original workflow satisfiability problem defined by Crampton and demonstrate that special cases of the problem are FTP.

5.1 More General Constraints

Crampton's model for workflows assumed that constraints had the form (ρ, s_1, s_2). In other words, constraints are applied to pairs of steps. Wang and Li then extended constraints so that they could have the form (ρ, S_1, S_2), where only one of S_1 and S_2 was required to be a singleton. It is easy to see that the equality relation is not interesting (at least for workflow satisfiability) for constraints of the original, simpler form. Indeed, WSP$(=)$ can be solved in polynomial time [17] for constraints of Crampton's form. However, the equality relation does have a significant impact on the complexity when the constraints may include non-singleton sets.

It is interesting to consider, therefore, the effect on WSP of having constraints of the form (ρ, S_1, S_2), where S_1 and S_2 are arbitrary subsets of S. The criterion for a plan to be valid remains the same: there should exist $s_i \in S_i$, $i = 1, 2$, such that $(\pi(s_1), \pi(s_2)) \in \rho$. For ease of reference, *Type* 1 constraints will refer to those of the form defined by Crampton [5], *Type* 2 will refer to Wang-Li constraints [17], and *Type* 3 will mean the most general form of constraints described above.

Theorem 5. *Let $\sim_1, \dots, \sim_\ell$ define an organizational hierarchy. Let $(S, U, \leqslant, A, C \cup C_\sim \cup C_{\not\sim})$ be a workflow, where C is the set of Type 2 constraints, $C_\sim$ is the set of Type 3 constraints of the form $(\sim_i, S_1, S_2)$ and $C_{\not\sim}$ is the set of Type 3 constraints of the form $(\not\sim_i, S_1, S_2)$. Then* WSP$(\sim_1, \dots, \sim_\ell, \not\sim_1, \dots, \not\sim_\ell)$ *can be solved in time*

$$\widetilde{O}((c + 2c')n2^{k+c'} + n^2 3^{k+c'}),$$

where $c = |C| + |C_{\not\sim}|$ and $c' = |C_\sim|$. Moreover, $c' \leqslant 3^k$, so WSP$(\sim_1, \dots, \sim_\ell, \not\sim_1, \dots, \not\sim_\ell)$ *is FPT.*

5.2 Ordered WSP

We note that the version of WSP considered so far in this paper makes no use of the order relation on the set of steps. This is a simplification introduced by Wang and Li [17]. In fact, the definition of workflow constraints in Crampton's model [5] prohibited constraints of the form (ρ, s, s') for $s > s'$. Moreover, a plan was required to specify an execution order for the steps in the workflow (in addition to the assignment of steps to users). This, in turn, means that Crampton's definition of constraint satisfaction (and hence of the workflow satisfiability problem) is more complex. More formally, we have the following definitions.

Definition 8. *Let $\mathcal{W} = (S, U, \leqslant, A, C)$ be a workflow comprising k steps. A tuple $(s_1, \dots, s_k)$ is an* execution schedule *for $\mathcal{W}$ if $\{s_1, \dots, s_k\} = S$ and, for all $1 \leqslant i < j \leqslant k$, $s_i \not\geqslant s_j$.*[7] *We say s_i* precedes *s_j in an execution schedule if $i < j$.*

For the workflow depicted in Figure 1, $(s_2, s_1, \dots)$ is not an execution schedule, for example, but $(s_1, s_2, s_3, s_5, s_4, s_6)$ and $(s_1, s_2, s_3, s_4, s_5, s_6)$ are.

Definition 9. *The (Type 1) constraint (ρ, s, s') is* satisfied *by execution schedule σ and plan π if one of the following holds: (i) s precedes s' in σ and $(\pi(s), \pi(s')) \in \rho$; (ii) s' precedes s in σ.*

The intuition here is that a constraint (ρ, s, s') is well-formed only if s could precede s' in the execution of some instance of the workflow (that is, either $s < s'$ or $s \parallel s'$). Moreover, if s does occur before s', then the execution of s' is constrained by ρ and the identity of the user that performed s. A modified version of WSP, based on the above definitions, is defined in the following way.

Ordered WSP (OWSP)
Input: A constrained workflow authorization schema $(S, U, \leqslant, A, C)$.
Output: True if there exists an execution schedule σ and a plan π that satisfy all constraints in C, and False otherwise.

Note that it may not be possible to find a valid plan π for a particular execution schedule σ. Conversely, there may be a plan π for which there exist schedules σ and σ' such that (σ, π) satisfies all constraints but (σ', π) does not.

Example 2. *Suppose $S = \{s_1, s_2, s_3, s_4\}$, such that $s_1 < s_2 < s_4$, $s_1 < s_3 < s_4$ and $s_2 \parallel s_3$, and $C = \{(\neq, s_2, s_3)\}$. Then we may define a plan π with the property that $\pi(s_2) = \pi(s_3)$, in which case $((s_1, s_2, s_3, s_4), \pi)$ is not a solution to* OWSP *(since there is a restriction on the user that performs*

[7] In other words, an execution schedule is a *linear extension* or *topological sort* of $(S, \leqslant)$.

s_3 once the identity of the user that performs s_2 has been fixed), whereas $((s_1, s_3, s_2, s_4), \pi)$ is a solution (since there is no restriction on the user that performs s_2 if s_3 is performed first).

In other words, there exist workflows for which a plan π is not a solution to WSP, but for which (σ, π) is a solution to OWSP for certain choices of σ. Crampton introduced the notion of a *well-formed workflow*, which has the following property: for all $s_i \parallel s_j$, $(\rho, s_i, s_j) \in C$ if and only if $(\tilde{\rho}, s_j, s_i) \in C$. To ensure that the workflow in the above example is well-formed, we would add the constraint $(\neq, s_3, s_2)$ to C. It is easy to see that OWSP for well-formed workflows and WSP are essentially equivalent, since a valid plan for one execution schedule will be a valid plan for any execution schedule [5, Lemma 9]. However, there are practical examples of workflows that are not well-formed [6].

Note that OWSP is only defined for Type 1 constraints (see Definition 9). Since WSP is shown to be W[1]-hard even if only Type 1 constraints are used [17] and since OWSP is a generalization of WSP provided only Type 1 constraints are used, OWSP is W[1]-hard. However, we can prove the following analog of Theorem 2. (We can also prove a similar analog of Theorem 4.)

Proposition 3. OWSP$(=, \neq)$ *is FPT.*

Proof. The crucial point is that the number of linear extensions of $(S, \leqslant)$ (and hence possible execution schedules) is determined only by k. Specifically, the number of linear extensions is no greater than $k!$. Then for each linear extension, we first eliminate any constraints that are rendered irrelevant by that linear extension and then solve an instance of WSP$(=, \neq)$ for the remaining constraints. Thus the complexity of OWSP$(=, \neq)$ is that of WSP$(=, \neq)$ multiplied by a function $f(k)$ dominated by $k!$. $\Box$

It is also worth noting that Crampton's constraints included the option of restricting the set of users to which the constraint applied. A constraint of the form (D, ρ, s_1, s_2) ($D \subseteq U$ is the *domain* of the constraint) is satisfied by a plan $\pi : S \to U$ if one of the following conditions holds: (i) $\pi(s_1) \notin D$; (ii) $(\pi(s_1), \pi(s_2)) \in \rho$. In other words, the constraint on the execution of s_2 only applies if s_1 is performed by a user in D; users in $U \setminus D$ are exempt. This allows us to specify that certain senior (trusted) users are not subject to separation-of-duty constraints, for example.

Some entries in the table indexed by users and subsets of S will be set to 1 rather than 0 with this extended form of constraints. It is easy to see, therefore, that the use of such constraints does not increase the complexity of WSP provided that we can test whether a user belongs to the domain of a constraint in time polynomial in n.

6. CONCLUDING REMARKS

6.1 Related Work

Work on computing plans for workflows that must simultaneously satisfy authorization policies and constraints goes back to the seminal paper of Bertino *et al.* [2]. This work considered linear workflows and noted the existence of an exponential algorithm for computing valid plans.

Crampton extended the model for workflows to partially ordered sets (equivalently, directed acyclic graphs) and to directed acyclic graphs with loops [5]. Wang and Li further extended this model to include Type 2 constraints and established the computational complexity and, significantly, the existence of fixed-parameter tractable algorithms for WSP$(=, \neq)$ [17]. Moreover, they established that WSP is W[1]-hard, in general.

Recent work by Basin *et al.* [1] introduces the notion of *release points* to model certain types of workflow patterns and defines the concept of *obstruction*, which is related to the notion of unsatisfiability. They prove that the *enforcement process existence problem* (EPEP), which is analogous to WSP for this extended notion of unsatisfiability, is NP-hard with complexity doubly-exponential in the number of users and constraints.

Independently of the work on authorization in workflows, there exists a vast literature on constraint satisfaction problems. In this context, Fellows *et al.* [10] studied WSP$(\neq)$ for Type 1 constraints and proved that this problem is fixed-parameter tractable.

Our work improves on that of Wang and Li and of Fellows *et al.* by establishing a tighter bound on the exponential factor of the fixed-parameter complexity for the relevant instances of WSP (Theorem 2). Moreover, our work establishes that it is unlikely that our bound can be significantly improved (Theorem 3). We extend the type of constraints that can be defined, introducing Type 3 constraints and demonstrating that WSP$(=, \neq)$ remains fixed-parameter tractable (Theorem 5). In addition, we extend the set of binary relations that can be used to specify authorization constraints. Specifically, we introduce a model for hierarchical organizations – suitable for organizations that are organized into mutually disjoint departments, or similar organizational units – and prove that WSP remains fixed-parameter tractable when such constraints are used (Theorem 4).

6.2 Future Work

There are many opportunities for further work in this area, both on the more theoretical complexity analysis and on extensions of WSP to richer forms of workflows. It is not hard to see that the function $f(k)$, in the complexity of the algorithm described in Proposition 3, equals $2^{O(k \log k)}$. Is there an algorithm such that the term in k has complexity $2^{O(k)}$? Note that such a reduction may be impossible to achieve (unless the Exponential Time Hypothesis fails), as in some problems considered by Lokshtanov *et al.* [13]. Does OWSP generalize in a natural way to Type 2 and Type 3 constraints? Are the resulting instances of OWSP$(=, \neq)$ FPT?

There exists a sizeable body of work on *workflow patterns.* Many workflows in practice require the ability to iterate a subset of steps in a workflow, or to branch (so-called OR-forks and AND-forks) and to then return to a single flow of execution (OR-joins and AND-joins) [16]. A variety of computational models and languages have been used to represent such workflows, including Petri nets and temporal logic. To our knowledge, the only complexity results for richer workflow patterns are those of Basin *et al.* described above, which can handle cycles. We will consider the fixed-parameter tractability of EPEP, and WSP for richer workflow patterns, in our future work.

A *parameterized problem* P can be represented as a relation $P \subseteq \Sigma^* \times \mathbb{N}$ over a finite alphabet Σ. In particular, WSP is a parameterized problem with parameter k, the number of steps. The idea of input reduction prior to solving the parameterized problem under consideration is captured in the following definition. Given a parameterized problem P, a *kernelization of* P is a polynomial-time algorithm that maps an instance (x, k) to an instance (x', k') such that (i) $(x, k) \in P$ if and only if $(x', k') \in P$ and (ii) $k' + |x'| \leq g(k)$ for some function g; (x', k') is the *kernel* and g is the *size* of the kernel. It is well-known and easy to prove that a decidable parameterized problem is FPT if and only if it has a kernel. It is easy to see that polynomial-size kernels will be particularly useful in practice. Unfortunately, many fixed-parameter tractable problems have no such kernels, unless NP $\subseteq$ coNP/poly (which is highly unlikely); see [4], for example.

We can show that, in general, WSP$(=, \neq)$ has no polynomial kernel unless NP $\subseteq$ coNP/poly. However, if all constraints in WSP$(=, \neq)$ involve only singletons, then WSP$(=, \neq)$ admits a kernel with at most k users. For a canonical organizational 3-hierarchy even the restriction to singletons does not help: there is no polynomial kernel unless NP $\subseteq$ coNP/poly. We will provide proofs of these results in a journal version of this paper.

7. REFERENCES

[1] D.A. Basin, S.J. Burri, and G. Karjoth. Obstruction-free authorization enforcement: Aligning security with business objectives. In *Proc. 24th IEEE Symp. on Comput. Sec. Foundations*, 99–113, 2011.

[2] E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.* 2(1): 65–104, 1999.

[3] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.* 39(2): 546–563, 2009.

[4] H.L. Bodlaender, R.G. Downey, M.R Fellows, and D. Hermelin, On problems without polynomial kernels. *J. Comput. Sys. Sci.* 75 (2009), 423–434.

[5] J. Crampton. A reference monitor for workflow systems with constrained task execution. In *Proc. 10th ACM Symp. on Access Control Models and Technologies* 38–47, 2005.

[6] J. Crampton and M. Huth. Synthesizing and verifying plans for constrained workflows: transferring tools from formal methods. In *Proc. 3rd ICAPS Workshop on Verification and Validation of Planning and Scheduling Systems*, 2011.

[7] R. Crowston, G. Gutin, M. Jones, V. Raman, and S. Saurabh. Parameterized complexity of MaxSat above average. *Proc. LATIN 2012*, Lect. Notes Comput. Sci. 7256 (2012), 184–194.

[8] R.G. Downey and M.R. Fellows. *Parameterized Complexity*, Springer Verlag, 1999.

[9] J. Flum and M. Grohe. *Parameterized Complexity Theory*, Springer Verlag, 2006.

[10] M. Fellows, T. Friedrich, D. Hermelin, N. Narodytska, and F. Rosamond. Constraint satisfaction problems: Convexity makes AllDifferent constraints tractable. In *Proc. 27th Intern. Joint Conf. on Artif. Intell.* 522-527, 2011.

[11] R. Impagliazzo, R. Paturi and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Sys. Sci.* 63(4): 512–530, 2001.

[12] T. Kaufman, M. Krivelevich, and D. Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM J. Comput.* 33(6): 1441–1483, 2004.

[13] D. Lokshtanov, D. Marx, and S. Saurabh. Slightly superexponential parameterized problems. In *Proceedings of SODA'2011*, 760–776, 2011.

[14] J. Moffett and E.C. Lupu. The uses of role hierarchies in access control. In *Proceedings of Fourth ACM Workshop on Role-Based Access Control*, 153–160, 1999.

[15] E. Tsang. *Foundations of Constraint Satisfaction*, Academic Press, 1993; available at `http://en.scientificcommons.org/43256499`.

[16] W.M.P. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros. Workflow patterns. *Distributed and Parallel Databases* 14(1): 5–51, 2003.

[17] Q. Wang and N. Li. Satisfiability and resiliency in workflow authorization systems. *ACM Trans. Inf. Syst. Secur.* 13(4): 1–35, 2010.

APPENDIX

In this appendix, we provide proofs of Theorems 3 and 5. Before proving Theorem 3, we define two problems related to 3-Sat and prove two preparatory lemmas.

c-Linear-3-Sat
Input: A 3-CNF formula ϕ with m clauses, and n variables such that $m \leq cn$, where c is a positive integer.
Output: Decide whether there is a truth assignment satisfying ϕ.

Let ϕ be a CNF formula. A truth assignment for ϕ is a *NAE-assignment* if, in each clause, it sets at least one literal true and at least one literal false. We say ϕ is *NAE-satisfiable* if there is a *NAE-assignment* for ϕ.

Not-All-Equal-3-Sat (NAE-3-Sat)
Input: A CNF formula ϕ in which every clause has exactly three literals.
Output: Decide whether ϕ is NAE-satisfiable.

The first of our lemmas, which we state without proof, is due to Impagliazzo *et al.* [11] (see also [7]).

Lemma 1. *Assuming the Exponential Time Hypothesis, there exist a positive integer L and a real number $\delta > 0$ such that L-*Linear-3-SAT *cannot be solved in time $O(2^{\delta n})$.*

Lemma 2. *Assuming the Exponential Time Hypothesis, there exists a real number $\epsilon > 0$ such that* NAE-3-SAT *with n variables cannot be solved in time $O(2^{\epsilon n})$, where n is the number of variables.*

Proof. Let L be an integer and δ be a positive real such that L-Linear-3-SAT cannot be solved in time $O(2^{\delta n})$. Such constants L and δ exist by Lemma 1. Suppose we have a polynomial time reduction from L-Linear-3-SAT to NAE-3-SAT and a positive integer c' such that if a formula in L-Linear-3-SAT has n variables then the corresponding formula in NAE-3-SAT has n' variables and $n' \leq c'n$. Let

$\epsilon = \delta/c'$ and suppose that NAE-3-SAT can be solved in time $O(2^{\epsilon n'})$, where n' is the number of variables. Then L-Linear-3-SAT can be solved in time $O(2^{\epsilon n'}) = O(2^{\delta n})$, a contradiction to the definition of δ.

It remains to describe the required polynomial time reduction from L-Linear-3-SAT to NAE-3-SAT. Recall that for every formula in L-Linear-3-SAT we have $m \leq Ln$, where m and n are the numbers of clauses and variables, respectively. We will show that our reduction gives $c' \leq 2(1+L)$. Let ϕ be a formula of L-Linear-3-SAT. Replace every clause $C = (u \vee v \vee w)$ in ϕ by

$$(u \vee v \vee x_C) \wedge (w \vee \overline{x_C} \vee y_C) \wedge (x_C \vee y_C \vee z) \qquad (1)$$

to obtain a formula ψ of NAE-3-SAT. Here variables x_C and y_C are new for every clause C and z is a new variable but it is common for all clauses of ϕ. We will show that ϕ is satisfiable if and only if ψ is NAE-satisfiable. This will give us $c'n \leq n + 2m + 1 \leq 2(1+L)n$ implying $c' \leq 2(1+L)$.

Let V_ϕ and V_ψ be the sets of variables of ϕ and ψ, respectively. Hereafter 1 stands for true and 0 for false.

Assume that ϕ is satisfiable and consider a truth assignment $\tau : V_\phi \to \{0,1\}$ that satisfies ϕ. We will extend τ to V_ψ such that the extended truth assignment is a NAE-assignment for ψ. We set $\tau(z) = 1$. For each clause $C = (u \vee v \vee w)$ of ϕ, we set $\tau(y_C) = 0$ and $\tau(x_C) = 1 - \max\{\tau(u), \tau(v)\}$. Consider (1). Since $\tau(y_C) = 0$ and $\tau(z) = 1$, τ is a NAE-assignment for the third clause in (1). Since $\max\{\tau(u), \tau(v)\} \neq \tau(x_C)$, τ is a NAE-assignment for the first clause of (1). Also, τ is a NAE-assignment for the second clause of (1) because either $\tau(x_C) = \tau(y_C) = 0$ or $\tau(u) = \tau(v) = 0$ and, hence, $\tau(w) = 1$.

Now assume that ψ is NAE-satisfiable and consider a NAE-assignment $\tau : V_\psi \to \{0,1\}$ for ψ. Since $\tau' : V_\psi \to \{0,1\}$ is a NAE-assignment for ψ if and only if so is $\tau''(t) = 1 - \tau'(t)$, $t \in V_\psi$, we may assume that $\tau(z) = 1$. Since τ is a NAE-assignment for the third clause of (1), we have $\min\{\tau(x_C), \tau(y_C)\} = 0$. If $\tau(x_C) = 0$ then $\max\{\tau(u), \tau(v)\} = 1$; otherwise $\tau(x_C) = 1$ and $\tau(y_C) = 0$ implying that $\tau(w) = 1$. Therefore, either $\max\{\tau(u), \tau(v)\} = 1$ or $\tau(w) = 1$ and, thus, C is satisfied by τ. □

Proof of Theorem 3. Consider a CNF formula ϕ, which is an instance of NAE-3-SAT. Let $\{s_1, \ldots, s_n\}$ be the variables of ϕ and let us denote the negation of s_i by s_{i+n} for each $i \in [n]$. For example, a clause $(s_1 \vee \overline{s_2} \vee \overline{s_3})$ will be written as $(s_1 \vee s_{n+2} \vee s_{n+3})$. For $j \in [2n]$, we write $s_j = 1$ if we assign true to s_j and $s_j = 0$, otherwise.

Now we construct an instance of WSP. The set of steps is $\{s_1, \ldots, s_k\}$, where $k = 2n$, and there are two users, u_0 and u_1. We will assign user u_i to a step s_j if and only if s_j is assigned i in ϕ. For each $j \in [n]$ we set constraint $(\neq, s_j, s_{j+n})$. For every clause of ϕ with literals s_ℓ, s_p, s_q we set constraint $(\neq, s_\ell, \{s_p, s_q\})$. We also assume that each user can perform every step subject to the above constraints.

Observe that the above instance of WSP is satisfiable if and only if ϕ is NAE-satisfiable. Thus, we have obtained a polynomial time reduction of NAE-3-SAT to WSP with $\neq$ being the only binary relation used in the workflow and with just two users. Now our theorem follows from Lemma 2. □

Proof of Theorem 5. The result follows from a very similar argument to that used in the proof of Theorem 4. Notice that our method for identifying ineligible sets for Type 2 constraints of the form $(\not\sim_i, s, S')$ works equally well for Type 3 constraints of the form $(\not\sim_i, S_1, S_2)$ (since a set F is ineligible if $S_1 \cup S_2 \subseteq F$).

However, we cannot use our method for constraints in $C_\sim$.[8] Nevertheless, we can rewrite the set of constraints in $C_\sim$ as Type 2 constraints, at the cost of introducing additional workflow steps. Specifically, we replace an Type 3 constraint $(\sim_i, S_1, S_2)$ with two Type 2 constraints, $(\sim_i, S_1, s_{\mathrm{new}})$ and $(\sim_i, s_{\mathrm{new}}, S_2)$, where s_{new} is a "dummy" step. Every user is authorized for s_{new}. This construction requires the replacement of c' Type 3 constraints by $2c'$ Type 2 constraints and the creation of c' new steps. Finally, we solve WSP for a workflow with n users, $k+c'$ steps and $c+2c'$ constraints, which has complexity $\widetilde{O}((c+2c')n2^{k+c'} + n^2 3^{k+c'})$.

We may assume without loss of generality that for all constraints of the form $(\sim_i, S_1, S_2)$ in $C_\sim$, $S_1 \cap S_2 = \emptyset$. (The constraint is trivially satisfied if there exists $s \in S_1 \cap S_2$, since we assume there exists at least one authorized user for every step.) Hence the number of constraints having this form is no greater than $\sum_{j=1}^{k} \binom{k}{j} 2^{k-j} = 3^k$. □

[8] Consider, for example, the Type 3 constraint $(=, \{s_1, s_2\}, \{s_3, s_4\})$: then (in the absence of other constraints) $\{s_i\}$ is eligible for $i \in [4]$; but any plan $\pi(s_i) = u_i$ such that $u_i = u_j$ if and only if $i = j$ does not satisfy the constraint (and is, therefore, invalid).