On the Parameterized Complexity and Kernelization of the Workflow Satisfiability Problem

JASON CRAMPTON, Royal Holloway, University of London
GREGORY GUTIN, Royal Holloway, University of London
ANDERS YEO, University of Johannesburg

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. Finally, we consider preprocessing for the problem and prove that in an important special case, in polynomial time, we can reduce the given input into an equivalent one, where the number of users is at most the number of steps. We also show that no such reduction exists for two natural extensions of this case, which bounds the number of users by a polynomial in the number of steps, provided a widely-accepted complexity-theoretical assumption holds.

1. INTRODUCTION

It is increasingly common for organizations to computerize their business and management processes. The co-ordination 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 steps that comprise a business process and the order in which those steps should be performed. Moreover, it is often the case that some form of access control, often role-based, should be applied to limit the execution of steps to authorized users. In addition, many workflows require controls on the users that perform groups of steps. The concept of a Chinese wall, for example, limits the set of steps that any one user can perform [Brewer and Nash 1989], as does separation-of-duty, which is a central part of the role-based access control model [American National Standards Institute 2004]. Hence, it is important that workflow management systems implement security controls that enforce authorization rules and business rules, in order to comply with statutory requirements or best practice [Basin et al. 2011]. It is these "security-aware" workflows that will be the focus of the remainder of this paper.

A simple, illustrative example for purchase order processing [Crampton 2005] 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 submit an invoice for the goods ordered, which is processed by the create payment step. When the supplier delivers the goods, a goods received note (GRN) must be signed and countersigned. Only then may the payment be approved and sent to the supplier. 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. In our example, 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. (We introduce more complex rules in Sections 2 and 5.)

(a) Ordering on steps   (b) Constraints

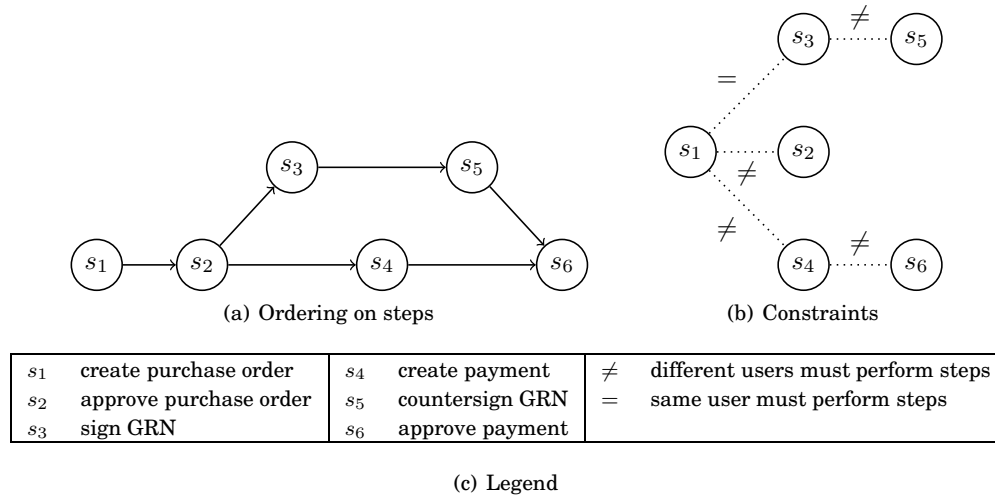| s_1 | create purchase order | s_4 | create payment | $\neq$ | different users must perform steps |
| s_2 | approve purchase order | s_5 | countersign GRN | $=$ | same user must perform steps |
| s_3 | sign GRN | s_6 | approve payment | | |

(c) Legend

Fig. 1.   A simple constrained workflow for purchase order processing

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 PROB-LEM (WSP) is known to be NP-hard, even when the set of constraints only includes constraints that have a relatively simple structure (and that would arise regularly in practice).[1]

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 [Wang and Li 2010]. One of the contributions 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. The use of cardinality constraints in access control policies has also attracted considerable interest in the academic community [Joshi et al. 2005; Sandhu et al. 1996; Simon and Zurko 1997]. Cardinality constraints can encode a number of useful requirements that cannot be encoded using the constraints that have been used in prior work on WSP. A second contribution of this paper is to introduce *counting* constraints for workflows—a natural extension of cardinality constraints—and to examine WSP when such constraints form part of a workflow specification.

Wang and Li [2010] 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 the parameterized complexity hypotheses FPT $\neq$ W[1] fails, which is highly unlikely. (We provide a short introduction to parameterized complexity in Section 3.1.) In this paper, we extend the results of Wang and Li in several different ways.

1. First, we introduce the notion of counting constraints, a generalization of cardinality constraints, and extend the analysis of WSP to include such constraints.
2. Our second contribution is to introduce a new approach to WSP, which makes use of a powerful, recent result in the area of exponential-time algorithms [Björklund et al. 2009]. We establish necessary and sufficient conditions on constraints that will admit the use of our approach. In particular, we show that counting constraints satisfy these conditions, as do the constraints considered by Wang and Li. This approach allows us to develop algorithms with a significantly better worst-case performance than those of Wang and Li. Moreover, we demonstrate that our result cannot be significantly improved, provided a well-known hypothesis about the complexity of solving 3-SAT holds.
3. Our third extension to the work of Wang and Li is to define constraints in terms of hierarchical organizational structures and to prove, using our new technique, that WSP remains fixed-parameter tractable in the presence of such hierarchical structures and hierarchy-related constraints.
4. Our fourth contribution is to instigate the systematic study of parameterized compression (also known as kernelization) of WSP instances.[2] We show that a result of Fellows et al. [2011, Theorem 3.3] on a problem equivalent to a special case of

---

[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 [Wang and Li 2010].
[2] Kernelization of WSP instances can be extremely useful in speeding up the solution of WSP: the compressed instance can be solved using any suitable algorithm (such as a SAT solver), not necessarily by an FPT algorithm.

WSP can be slightly extended and significantly improved using graph matchings. We also prove that two natural further extensions of the result of Fellows et al. are impossible subject to a widely-accepted complexity-theoretical hypothesis.

In the next section, we introduce the workflow satisfiability problem. In Section 3, we provide a brief introduction to fixed-parameter tractability, prove a general result characterizing the constraints for which WSP is fixed-parameter tractable, and apply this result to counting constraints. In Section 4 we extend the results of Wang and Li, by improving the complexity of the algorithms used to solve WSP and by introducing constraints based on equivalence relations. In Section 5, we introduce a model for an organizational hierarchy and a class of constraint relations defined in terms of such hierarchies. We demonstrate that WSP remains fixed-parameter tractable for workflow specifications that include constraints defined over an organizational hierarchy. In Section 6, we discuss kernelization of WSP and prove that in an important special case, in polynomial time, we can transform the given input into an equivalent one, where the number of users is at most the number of steps. We also show that no polynomial transformation exists for two natural extensions of this case, which bounds the number of users by a polynomial in the number of steps, unless a certain complexity-theoretical assumption fails. The paper concludes with a summary of our contributions and discussions of related and future work.

2. THE WORKFLOW SATISFIABILITY PROBLEM

In this section, we introduce our notation and definitions, derived from earlier work by Crampton [2005] and Wang and Li [2010], and then define the workflow satisfiability problem.

A partially ordered set (or poset) is a pair $(X, \leqslant)$, where $\leqslant$ is a reflexive, antisymmetric and transitive binary relation defined over X. If $(X, \leqslant)$ is a poset, then we write $x \parallel y$ if x and y are incomparable; that is, $x \not\leqslant y$ and $y \not\leqslant x$. 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 $\{1, \ldots, n\}$.

Definition 2.1. 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. Our definition of workflow specification does not permit repetition of tasks (loops) or repetition of sub-workflows (cycles). User u is authorized to perform step s only if $(s, u) \in A$.[3] We assume that for every step $s \in S$ there exists some user $u \in U$ such that $(s, u) \in A$.

Definition 2.2. 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 $s \in S$.

The access control policy embodied in the authorization relation A imposes restrictions on the users that can perform specific steps in the workflow. A *workflow authorization constraint* imposes restrictions on the execution of *sets* of steps in a workflow.

---

[3]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, R^2BAC – the role-and-relation-based access control model of Wang and Li [2010] – 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 R^2BAC), it is straightforward to derive A. We prefer to use A in order to simplify the exposition.

A constraint is defined by some suitable syntax and its meaning is provided by the restrictions the constraint imposes on the users that execute the sets of steps defined in the constraint. In other words, constraint satisfaction is defined with reference to a plan; a *valid* plan is one that is authorized and allocates users in such a way that the constraint is satisfied. A very simple example of a constraint is one requiring that steps s and s' are executed by different users. Then a valid plan π (with respect to this constraint) has the property that $\pi(s) \neq \pi(s')$. A *constrained workflow authorization schema* is a tuple $(S, U, \leqslant, A, C)$, where C is a set of workflow constraints.[4] A plan is *valid* for an authorization schema if it is authorized and satisfies all constraints in C. We define particular types of constraints in Section 2.2 and 2.3.

We may now define the workflow satisfiability problem, as defined by Wang and Li [2010].

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

---

We will write c, n and k to denote the number of constraints, users and steps, respectively, in an instance of WSP. We will analyze the complexity of the workflow satisfiability problem in terms of these parameters.

2.1. Applications of WSP

An algorithm that solves WSP can be used by a workflow management system in one of three ways, depending on how users are allocated to steps in an instance of the workflow. Some systems allocate an authorized user to each step when a workflow instance is generated. Other systems allocate users to only those steps that are ready to be performed in an instance of the workflow. (A step is ready only if all its immediate predecessor steps have been completed.) The third possibility is to allow users to select a step to execute from a pool of ready steps maintained by the workflow management system.

For the first type of system, it is important to know that a workflow is satisfiable and an algorithm that solves WSP can simply be used as a static analysis tool. 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.

For the second and third cases, the system must guarantee that the choice of user to execute a step (whether it is allocated by the system or selected by the user) does not prevent the workflow instance from completing. This analysis needs to be performed each time a user is allocated to, or selects, a step in a workflow instance. The question can be resolved by solving a new instance of WSP, in which those steps to which users have been allocated are assumed to have a single authorized user (namely, the user that has been allocated to the task) [Crampton 2005, §3.2]. Assuming that these checks should incur as little delay as possible, particularly in the case when users select steps in real time [Kohler and Schaad 2008], it becomes even more important to find an algorithm that can decide WSP as efficiently as possible.

The definition of workflow satisfiability given above assumes that the set of users and the authorization relation are given. This notion of satisfiability is appropriate when the workflow schema is designed "in-house". A number of large information technology companies develop business process systems which are then configured by the

---

[4]The set of constraints defines what has been called a *history-dependent authorization policy* [Basin et al. 2012]; the relation A defines a *history-independent policy*.

end users of those systems. Part of that configuration includes the assignment of users to steps in workflow schemas. The developer of such a schema may wish to be assured that the schema is satisfiable for some set of users and some authorization relation, since the schema is of no practical use if no such user set and authorization relation exist. The desired assurance can be provided by solving an instance of WSP in which there are k users, each of which is authorized for all steps. The developer may also determine the minimum number of users required for a workflow schema to be satisfiable. The minimum number must be between 1 and k and, using a binary search, can be determined by examining $\lceil \log_2 k \rceil$ instances of WSP.

2.2. Constraint Types

In this paper, we consider two forms of constraint: *counting* constraints and *entailment* constraints. A counting constraint has the form (t_ℓ, t_r, S'), where $1 \leqslant t_\ell \leqslant t_r \leqslant k$ and $S' \subseteq S$. A counting constraint is a generalization of the cardinality constraints introduced in the RBAC96 model [Sandhu et al. 1996] and widely adopted by subsequent access control models [American National Standards Institute 2004; Bertino et al. 2001; Joshi et al. 2005].

A plan $\pi : S \to L$ satisfies counting constraint (t_ℓ, t_r, S') if a user performs either no steps in S' or between t_ℓ and t_r steps. In other words, no user is assigned to more than t_r steps in S' and each user (if involved in the execution of steps in S') must perform at least t_ℓ steps. Many requirements give rise to counting constraints of the form (t, t, S'), which we will abbreviate to (t, S'). A number of requirements that arise in the literature and in practice can be represented by counting constraints.

Separation of duty. The constraint $(1, \{s', s''\})$ requires that no user executes both s' and s''. More generally, the constraint $(1, |S'| - 1, S')$ requires that no user executes all the steps in S'.

Binding of duty. The constraint $(2, \{s', s''\})$ requires that the same user executes both s' and s''. More generally, the constraint $(|S'|, S')$ requires that all steps in S' are executed by the same user.

Division of duty. The constraint $(\lfloor |S'|/v \rfloor, \lceil |S'|/v \rceil, S')$ requires that the steps in S' are split as equally as possible between v different users. The special case $(1, S')$ requires that a different user performs each step in S'.

Threshold constraints. The constraint $(1, t, S')$ requires that no user executes more than t steps in S'.[5]

Generalized threshold constraints. The constraint (t_ℓ, t_r, S') requires that each user (involved in the execution of steps in S') performs between t_ℓ and t_r of those steps.

Counting constraints are not able to encode certain types of requirements. For this reason, we also consider entailment constraints, which have the form (ρ, S', S''), where $\rho \subseteq U \times U$ and $S', S'' \subseteq S$. A plan π satisfies entailment constraint (ρ, S', S'') if and only if there exists $s' \in S'$ and $s'' \in S''$ such that $(\pi(s'), \pi(s'')) \in \rho$. A plan π satisfies a set of constraints C (which may be a mixture of counting and entailment constraints) if π satisfies each constraint in C.

Counting constraints represent "universal" restrictions on the execution of steps (in the sense that every user in a plan must satisfy the requirement stipulated). In contrast, entailment constraints are "existential" in nature: they require the existence of

---

[5]These constraints are similar in structure and analogous in meaning to SMER (statically, mutually-exclusive, role) constraints [Li et al. 2007]; the SMER constraint (t, S') requires that no user is authorized for t or more roles in the set of roles S'. These constraints are also similar to the cardinality constraints defined in RBAC96 [Sandhu et al. 1996].

a pair of steps for which a condition on the two users who execute those steps (defined by the binary relation ρ) is satisfied.

We could write δ to the denote the diagonal relation $\{(u,u) : u \in U\}$ and $\overline{\delta}$ to denote $(U \times U) \setminus \delta$. However, we will prefer to use the less formal, but more intuitive, notation $(\neq, S', S'')$ and $(=, S', S'')$ to denote the constraints $(\overline{\delta}, S', S'')$ and (δ, S', S''), respectively.

There are some requirements that can be represented by a counting constraint or an entailment constraint. The counting constraint $(1, \{s_1, s_2\})$, for example, is satisfied by plan π if and only if the entailment constraint $(\neq, \{s_1\}, \{s_2\})$ is satisfied. We say that two constraints γ and γ' are *equivalent* if a plan π satisfies γ if and only if it satisfies γ'. Thus $(1, \{s_1, s_2\})$ is equivalent to $(\neq, \{s_1\}, \{s_2\})$. Similarly, $(2, \{s_1, s_2\})$ is equivalent to $(=, \{s_1\}, \{s_2\})$. Nevertheless, there is no counting constraint (or set of such constraints) that is equivalent to $(=, S_1, S_2)$. Equally, there is no entailment constraint (or set of such constraints) that is equivalent to (t, S').

2.3. Entailment Constraint Subtypes

Previous work on workflow satisfiability has not considered counting constraints. Moreover, our definition of entailment constraint is more general than prior definitions. Thus, we study more general constraints for WSP than have been investigated before.

Crampton [2005] defined entailment constraints in which S_1 and S_2 are singleton sets: we will refer to constraints of this form as *Type* 1 constraints; for brevity we will write (ρ, s_1, s_2) for the Type 1 constraint $(\rho, \{s_1\}, \{s_2\})$. Wang and Li [2010] defined constraints in which at least one of S_1 and S_2 is a singleton set: we will refer to constraints of this form as *Type* 2 constraints and we will write (ρ, s_1, S_2) in preference to $(\rho, \{s_1\}, S_2)$. The Type 2 constraint (ρ, s_1, S_2) is equivalent to (ρ, S_2, s_1) if ρ is symmetric, in which case we will write (ρ, s_1, S_2) in preference to (ρ, S_2, s_1). Note that both δ and $\overline{\delta}$ are symmetric binary relations. Constraints in which S_1 and S_2 are arbitrary sets will be called *Type* 3 constraints.

We note that Type 1 constraints can express requirements of the form described in Section 1, where we wish to restrict the combinations of users that perform pairs of steps. The plan π satisfies constraint $(=, s, s')$, for example, if the same user is assigned to both steps by π, and satisfies constraint $(\neq, s, s')$ if different users are assigned to s and s'.

Type 2 constraints provide greater flexibility, although Wang and Li, who introduced these constraints, do not provide a use case for which such a constraint would be needed. However, there are forms of separation-of-duty requirements that are most naturally encoded using Type 3 constraints. Consider, for example, the requirement that a set of steps $S' \subseteq S$ must not all be performed by the same user [Armando et al. 2009]. We may encode this as the constraint $(\neq, S', S')$, which is satisfied by a plan π only if there exists two steps in S' that are allocated to different users by π.[6] The binding-of-duty constraint $(=, S', S'')$ cannot be directly encoded using Type 2 constraints or counting constraints.

Now consider a business rule of the form "two steps must be performed by members of the same organizational unit". The constraint relations $=$ and $\neq$ do not allow us to define such constraints. In Section 4, we model constraints of this form using

---

[6]It is interesting to note that a Type 3 constraint $(\neq, S', S'')$ can be encoded as a Type 2 constraint, thereby providing retrospective motivation for the introduction of Type 2 constraints by Wang and Li. In particular, we may encode $(\neq, S', S'')$ as $(\neq, s, S' \cup S'' \setminus \{s\})$ for some $s \in S' \cup S''$. The equivalence of these two constraints is left as an exercise for the interested reader. (Note that we may also encode this requirement as the counting constraint $(1, |S'| - 1, S')$.)

equivalence relations defined on the set of users. In Section 5, we introduce a model for hierarchical organizational structures, represented in terms of multiple, related equivalence relations defined on the set of users. We then consider constraints derived from such equivalence relations and the complexity of WSP in the presence of such constraints.

Henceforth, we will write $\mathrm{WSP}(\rho_1, \ldots, \rho_t)$ to denote a special case of WSP in which all constraints have the form (ρ_i, S', S'') for some $\rho_i \in \{\rho_1, \ldots, \rho_t\}$ and for some $S', S'' \subseteq S$. We will write $\mathrm{WSP}_i(\rho_1, \ldots, \rho_t)$ to denote a special case of $\mathrm{WSP}(\rho_1, \ldots, \rho_t)$, in which there are no constraints of Type j for $j > i$. So $\mathrm{WSP}_1(=, \neq)$, for example, indicates an instance of WSP in which all constraints are of Type 1 and only includes constraints of the form $(=, s_1, s_2)$ or $(\neq, s_1, s_2)$ for some $s_1, s_2 \in S$. For ease of exposition, we will consider counting constraints and entailment constraints separately. Our results, however, hold when a workflow specification includes both types of constraints.

3. WSP **AND FIXED-PARAMETER TRACTABILITY**

In order to make the paper self-contained, we first provide a short overview of parameterized complexity, what it means for a problem to be fixed-parameter tractable, and summarize the results obtained by Wang and Li for WSP. We then introduce the notion of an eligible set of steps. The identification of eligible sets is central to our method for solving WSP. In the final part of this section, we state and prove a "master" theorem from which a number of useful results follow as corollaries. The master theorem also provides useful insights into the structure of constraints that will result in instances of WSP that are fixed-parameter tractable.

3.1. 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 (worst-case) 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 $\mathrm{WSP}(\neq)$ [Wang and Li 2010, Lemma 3]. In short, WSP is hard to solve in general. The importance of finding an efficient algorithm for solving WSP led Wang and Li to look at the problem from the perspective of parameterized complexity [Wang and Li 2010, §4].

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, using an elementary argument, that $\mathrm{WSP}_2(\neq)$ is FPT and can be solved in time $O(k^{k+1}N)$, where N is the size of the entire input to the problem [Wang and Li 2010, Lemma 8]. They also showed that $\mathrm{WSP}_2(\neq, =)$ is FPT [Wang and Li 2010, Theorem 9], using a rather more complex approach: specifically, they constructed an algorithm that runs in time $O(k^{k+1}(k-1)^{k2^{k-1}}N)$; it follows that $\mathrm{WSP}_2(=, \neq)$ is 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 collection of parameterized complexity classes, $\mathrm{W}[1], \mathrm{W}[2], \ldots$, with FPT $\subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{XP}$. Informally, a parameterized problem belongs to the complexity class $\mathrm{W}[i]$ if there exists an FPT algorithm that transforms every

instance of the problem into an instance of WEIGHTED CIRCUIT SATISFIABILITY for a circuit of weft i. It can be shown that FPT is the class W[0]. The problems INDE-PENDENT SET and DOMINATING SET are in W[1] and W[2], respectively. It is widely-believed and often assumed that FPT $\neq$ W[1]. For a more formal introduction to the W family of complexity classes, see Flum and Grohe [2006].

Wang and Li [2010, Theorem 10] proved that WSP (for arbitrary relations defined on the user set) is W[1]-hard in general, using a reduction from INDEPEN-DENT SET. By definition, FPT is a subset of W[1] and a parameterized analog of Cook's Theorem [Downey and Fellows 1999] as well as the Exponential Time Hypothesis [Flum and Grohe 2006; Impagliazzo et al. 2001] 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.

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, Björklund et al. [2009] and Kaufman et al. [2004]—to avoid cumbersome runtime bounds.

3.2. Eligible Sets

The basic idea behind our results 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 equiv-alence relation $\sim_\pi$ on S, where $s \sim_\pi s'$ if and only if $\pi(s) = \pi(s')$. We denote the set of equivalence classes of $\sim_\pi$ by S/π and write $[s]_\pi$ to denote the equivalence class con-taining s. An equivalence class in S/π comprises the set of steps that are assigned to a single user by plan π. It is easy to see that there are certain "forbidden" subsets S' of S for which there cannot exist a valid plan π such that $S' \in S/\pi$. Consider, for example, the constraint $(\neq, s, s')$: 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 $\{s, s'\} \in S/\pi$. This motivates the following definition.

Definition 3.1. Given a workflow $(S, U, \leqslant, A, C)$ and a constraint $\gamma \in C$, a set $F \subseteq S$ is γ-*ineligible* if any plan $\pi : S \to U$ such that $F \in S/\pi$ violates γ. We say F is *eligible* if and only if it is not ineligible. We say $F \subseteq S$ is C-*ineligible* or simply *ineligible* if F is γ-ineligible for some $\gamma \in C$.

A necessary condition for a valid plan is that no equivalence class is an ineligible set; equivalently, every equivalence class in a plan must be an eligible set. For many constraints γ, we can determine whether $F \subseteq S$ is γ-ineligible or not in time polynomial in the number of steps. Consider, for example, the requirement that no user executes more than t steps: then $F \subseteq S$ is eligible if and only if $|F| \leqslant t$. Similarly, we can test for the ineligibility of F with respect to $(\neq, \{s_1, s_2\})$ by determining whether $F \supseteq \{s_1, s_2\}$.

Definition 3.2. We say a constraint γ is *regular* if any plan π in which each equiva-lence class $[s]_\pi$ is an eligible set satisfies γ.

The regularity of a constraint is a sufficient condition to guarantee that we can con-struct a valid plan using eligible sets. With one exception, all constraints we consider are regular.

PROPOSITION 3.3. *All counting constraints are regular and all entailment con-straints of the form $(\neq, S_1, S_2)$ are regular. Entailment constraints of the form $(=, S_1, S_2)$ are regular if at least one of S_1 and S_2 is a singleton set.*

PROOF. The result is trivial for counting constraints.

Given an entailment constraint $(\neq, S_1, S_2)$, a plan π in which all equivalence classes are eligible, and $[s]_\pi$ for some $s \in S_1 \cup S_2$, we have that $[s]_\pi \not\supseteq S_1 \cup S_2$ (since, by assumption, $[s]_\pi$ is eligible). Hence, there exists an element $s' \in S_1 \cup S_2$ with $s' \notin [s]_\pi$. Since the equivalence classes in S/π form a partition of S, there exists an equivalence class $[s']_\pi \neq [s]_\pi$. Hence, the constraint is satisfied (since each equivalence class is assigned to a different user). Thus the constraint is regular.

We demonstrate, by exhibiting a counterexample, that a partition of S into eligible sets does not guarantee the satisfaction of a Type 3 constraint of the form $(=, S_1, S_2)$. Consider, for example, $S = \{s_1, s_2, s_3, s_4\}$ and the constraint $(=, \{s_1, s_2\}, \{s_3, s_4\})$. Then $\{s_1\}, \ldots, \{s_4\}$ are eligible sets, but a plan in which u_i is assigned to s_i is not valid.

Finally, consider the Type 2 constraint $(=, s_1, S_2)$. Any eligible set for this constraint that contains s_1 must contain an element of S_2. Hence a partition of S into eligible sets ensures that the constraint will be satisfied (and hence that the constraint is regular). □

3.3. Reducing WSP to MAX WEIGHTED PARTITION

We now state and prove our main result. We believe this result subsumes existing results in the literature on the complexity of WSP. Moreover, the result considerably enhances our understanding of the types of constraints that can be used in a workflow specification if we wish to preserve fixed-parameter tractability of WSP. We explore the consequences and applications of our result in Sections 4 and 5.

THEOREM 3.4. *Let $W = (S, U, \leqslant, A, C)$ be a workflow specification such that (i) each constraint γ is regular and (ii) there exists an algorithm that can determine whether $F \subseteq S$ is γ-eligible in time polynomial in k. Then the workflow satisfiability problem for W can be solved in time $\widetilde{O}(2^k(c + n^2))$.*

The proof of this result reduces an instance of WSP to an instance of the MAX WEIGHTED PARTITION problem, which, by a result of Björklund et al. [2009], is FPT. We state the problem and the relevant result, before proving Theorem 3.4.

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 3.5 (BJÖRKLUND ET AL. [2009]). MAX WEIGHTED PARTITION *can be solved in time $\widetilde{O}(2^k n^2 M)$.*

PROOF OF THEOREM 3.4. We construct a binary matrix with n rows (indexed by elements of U) and 2^k columns (indexed by elements of 2^S): every entry in the column labeled by the empty set is defined to be 1; the entry indexed by $u \in U$ and $F \subseteq S$ is defined to be 0 if and only if $F \neq \emptyset$ is C-ineligible or there exists $s \in F$ such that $(s, u) \notin A$. In other words, the non-zero matrix entry indexed by u and F defines a C-eligible set and u is authorized for all steps in F, and thus 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 $\{\phi_u\}_{u \in U}$. Given that $\phi_u(F) \leqslant 1$, $\sum_{u \in U} \phi_u(F_u) \leqslant n$, with equality if and only if we can partition S into different C-eligible blocks and assigned them to different users. Since each γ is regular, W is satisfiable if and only if MWP returns a partition having weight n.

We now consider the complexity of the above algorithm. By assumption, we can identify the ineligible sets in $O(c \cdot k^d \cdot 2^k) = \widetilde{O}(c2^k)$ time for some integer d independent of k and c. And we can check whether a user is authorized for all steps in $F \subseteq S$ in $O(k)$ time. Thus we can construct the matrix in $O(2^k \cdot n \cdot k) = \widetilde{O}(2^k n)$ time. Finally, we can solve MAX WEIGHTED PARTITION in $\widetilde{O}(2^k n^2)$ time. Thus, the total time required to solve WSP for W is $\widetilde{O}(2^k(c + n + n^2)) = \widetilde{O}(2^k(c + n^2))$.   □

THEOREM 3.6. *WSP is FPT for any workflow specification in which all the constraints are counting constraints.*

PROOF. A plan $\pi : S \to L$ satisfies counting constraint $\gamma = (t_\ell, t_r, S')$ if a user performs either no steps in S' or between t_ℓ and t_r steps. Hence, $F \subseteq S$ is eligible if and only if $t_\ell \leqslant |F| \leqslant t_r$, a test that can clearly be evaluated in $O(k)$ time. The result now follows by Proposition 3.3 and Theorem 3.4.   □

While the above result appears easy to state and prove, nothing was known about the complexity of incorporating such constraints into workflow specifications. Moreover, counting constraints can be used to encode (Type 1) entailment constraints of the form $(\neq, s_1, s_2)$ and $\mathrm{WSP}_1(\neq)$ is known to be NP-complete [Wang and Li 2010, Lemma 3]. Finally, counting constraints can encode requirements that cannot be expressed using entailment constraints. Hence, WSP in the presence of counting constraints is at least as hard as $\mathrm{WSP}_1(\neq)$. Therefore, there is no immediate reason to suppose that WSP for counting constraints would be FPT. In short, Theorem 3.6 is non-trivial, thus demonstrating the power of Theorem 3.4.

At first glance, it is perhaps surprising to discover that counting constraints have no effect on the fixed-parameter tractability of WSP. However, on further reflection, the structure of the proof of Theorem 3.4 suggests that *any* constraint whose satisfaction is phrased in terms of the steps that a single user performs can be incorporated into a workflow specification without comprising fixed-parameter tractability.

It also becomes apparent that there are certain constraints whose inclusion may cause problems. Any constraint whose satisfaction is defined in terms of the set of users that perform a set of steps may be problematic. The requirement that a workflow be performed by at least three users, for example, cannot be encoded using the counting or entailment constraints we have defined in this paper. Moreover, it is difficult to envisage an eligibility test for such a constraint and, if such a test exists, whether it can be evaluated in time polynomial in k. However, we can express a constraint of this form as a counting constraint such that the original constraint is satisfied if the counting constraint is satisfied. Specifically, the requirement that a set of S' steps be performed by at least t users can be enforced by ensuring that each user performs no more than $(|S'| - 1)/(t - 1)$ steps.[7]

4. ENTAILMENT CONSTRAINTS

In this section we focus on workflow specifications that include only entailment constraints. In doing so, we demonstrate further the power of Theorem 3.4. We also show that the time complexity obtained in Theorem 3.4 cannot be significantly improved even for a very special case of WSP. We conclude with a discussion of and comparison with related work.

---

[7]Of course, this means that certain plans that do not violate the original requirement are invalid. That is, the counting constraint "over-enforces" the original requirement. See the work of Li et al. [2007] for further details on constraint rewriting of this nature.

4.1. WSP($\neq$)

By Proposition 3.3, any constraint γ of the form $(\neq, S_1, S_2)$ is regular. Moreover, there exists an easy test to determine whether $F \subseteq S$ is γ-ineligible. Specifically, F is γ-ineligible if and only if $F \supseteq S_1 \cup S_2$, since any plan that allocated a single user to the steps in F would be invalid. Hence, we can determine in time polynomial in the sizes of F, S_1 and S_2 (that is, in k) the eligibility of γ.

THEOREM 4.1. *WSP($\neq$) can be solved in time $\widetilde{O}(2^k(c + n^2))$.*

PROOF. The result follows from Theorem 3.4 and the fact that every constraint is regular and the eligibility of any constraint can be determined in time polynomial in k. □

Our next result asserts that it is impossible, assuming the well-known *Exponential Time Hypothesis* [Impagliazzo et al. 2001], 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 4.2. *Even if there are just two users, WSP$_2$($\neq$) 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 can be found in the appendix.

4.2. WSP($=$)

Given a constraint γ of the form $(=, S_1, S_2)$, any set F that contains S_1 but no element of S_2 is ineligible; equally, any set F that contains S_2 but no element of S_1 is ineligible. Hence, we can determine γ-ineligibility in time polynomial in k (as we only require subset inclusion and intersection operations on sets whose cardinalities are no greater than k). However, a constraint γ of the form $(=, S_1, S_2)$ is not necessarily regular (Proposition 3.3). Nevertheless, we have the following result.

THEOREM 4.3. *WSP$_2$($=$) 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. WSP($=$) can be solved in time $\widetilde{O}(2^{k+c}(c + n^2))$.*

PROOF. The first result follows immediately from Theorem 3.4 and Proposition 3.3, since the latter result asserts that constraints of the form $(=, s_1, S_2)$ are regular.

To obtain the second result, we rewrite a Type 3 constraint $(=, S_1, S_2)$ as two Type 2 constraints, at the cost of introducing additional workflow steps. Specifically, we replace a Type 3 constraint $(=, S_1, S_2)$ with the constraints $(=, S_1, s_{\text{new}})$ and $(=, s_{\text{new}}, S_2)$, where s_{new} is a "dummy" step. Every user is authorized for s_{new}. Observe that if we have a plan that satisfies $(=, S_1, S_2)$ then there exists a user u and steps $s_1 \in S_1$ and $s_2 \in S_2$ such that $\pi(s_1) = \pi(s_2)$. Hence we can find a plan that satisfies $(=, S_1, s_{\text{new}})$ and $(=, s_{\text{new}}, S_2)$: specifically, we extend π by defining $\pi(s_{\text{new}}) = u$. Similarly, if we have a plan that satisfies $(=, S_1, s_{\text{new}})$ and $(=, s_{\text{new}}, S_2)$ then there exists a user u and steps s_1 and s_2 such that $u = \pi(s_{\text{new}}) = \pi(s_1) = \pi(s_2)$ and we may construct a valid plan for $(=, S_1, S_2)$.

The rewriting of a (Type 3) constraint $(=, S_1, S_2)$ requires the replacement of one Type 3 constraint with two Type 2 constraints and the creation of one new step. In other words, we can derive an equivalent instance of WSP$_2$($=$) having no more than c

additional constraints and no more than c additional steps. Since Type 2 constraints are regular, the result now follows by Theorem 4.1. □

COROLLARY 4.4. WSP($=$) *is FPT.*

PROOF. We may assume without loss of generality that $S_1 \cap S_2 = \emptyset$: 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 $\sum_{j=1}^{k} \binom{k}{j} 2^{k-j} = 3^k$. Hence, WSP($=$) is FPT, since we can replace 2^{k+c} in the run-time by 2^{k+3^k}, as required. □

4.3. WSP($=, \neq$) and Related Work

We can combine the results of the previous sections in a single theorem. Clearly, we could also incorporate counting constraints into this result.

THEOREM 4.5. WSP($=, \neq$) *can be solved in time* $\widetilde{O}(2^{k+c}(c + n^2))$.

The special case of the workflow satisfiability problem $\text{WSP}_2(\neq)$ 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 [Wang and Li 2010, Lemma 8]. Fellows et al. [2011] considered the fixed-parameter tractability of a special case of the *constraint satisfaction problem* [Tsang 1993] in which all constraints have the same form; with these restrictions, the constraint satisfaction problem is identical to $\text{WSP}_1(\neq)$. The algorithm of Fellows et al. has complexity $O(k!kn) = 2^{O(k \log k)}n$, where n is the number of users [Fellows et al. 2011, Theorem 3.1]. Our algorithm has complexity $\widetilde{O}(2^k(c+n^2)) = O(2^{k+d \log k}(c + n^2))$, where $d = O(1)$, which represents a considerable improvement in the term in k.

More significantly, Wang and Li [2010, Theorem 9] showed that $\text{WSP}_2(\neq, =)$ is FPT; the complexity of their algorithm is $O(k^{k+1}(k-1)^{k2^{k-1}}n)$. Our algorithm to solve $\text{WSP}_2(=, \neq)$ retains the complexity $\widetilde{O}(2^k(c + n^2))$, which is clearly a substantial improvement on the result of Wang and Li. Finally, we note that our results are the first to consider Type 3 constraints.

4.4. Constraints Based on Equivalence Relations

The work of Crampton [2005, §2] and of Wang and Li [2010, Examples 1, 2] has noted that a constraint of practical interest is that users performing two steps must be from the same department.[8] In the workflow illustrated in Figure 1 one might require, for example, 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 be different. More generally, 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, there are many practical situations in which some auxiliary information defines an equivalence relation on the set of users (membership of department, for example) where we may wish to require that two steps are performed by users belonging to either the same equivalence class or to different equivalence classes. 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.

---

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

Given an equivalence relation $\sim$ on U, a plan π satisfies the constraint $(\sim, S_1, S_2)$ 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 the same equivalence class. Similarly, a plan π satisfies the constraint $(\not\sim, S_1, S_2)$ 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 equivalence classes. Hence, the constraint $(\sim, s_3, s_5)$ would encode 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). More generally, a constraint of the form $(\sim, s, s')$ represents a weaker constraint than one of the form $(=, s, s')$, since more plans satisfy such a constraint. Conversely, a constraint of the form $(\not\sim, s, s')$ is stronger than $(\neq, s, s')$, as it requires that the two users who perform s and s' are different and, in addition, they belong to different equivalence classes.

THEOREM 4.6. *For any user set U and any equivalence relation $\sim$ defined on U, $\mathrm{WSP}(\sim, \not\sim)$ 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'.
Observe that $\mathcal{W}$ is satisfiable if and only if $\mathcal{W}'$ is, and deciding the satisfiability of $\mathcal{W}'$ is FPT by Theorem 4.1 and Corollary 4.4. □

Of course, we could also include counting constraints in the workflow specification. Let us assume, for ease of explanation, that an equivalence relation partitions a user set into different organizational units.

Separation of duty. The constraint $(1, \{s', s''\})$ requires that users from different organizational units perform s' and s''. More generally, the constraint $(1, |S'| - 1, S')$ requires that no single unit executes all the steps in S'.

Binding of duty. The constraint $(2, \{s', s''\})$ requires that users from the same organizational unit execute both s' and s''. More generally, the constraint $(|S'|, S')$ requires that all steps in S' are executed by users from the same unit.

The other forms of counting constraints introduced in Section 2.2 can be interpreted in analogous ways in the presence of an equivalence relation defined on the set of users.

5. ORGANIZATIONAL HIERARCHIES

We now show how we can use multiple equivalence relations to define an organizational hierarchy. In Section 5.2, 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 $(F_1, \ldots, F_n)$ such that $F_1 \cup \cdots \cup F_n = S$ 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*.[9]

Definition 5.1. Let $(X_1, \ldots, X_p)$ and $(Y_1, \ldots Y_q)$ be p- and q-partitions of the same set. We say that $(Y_1, \ldots Y_q)$ is a *refinement* of $(X_1, \ldots, X_p)$ if for each $i \in [q]$ there exists $j \in [p]$ such that $Y_i \subseteq X_j$.

---

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

Definition 5.2. Let U be the set of users in an organization. An *organizational ℓ-hierarchy* is a collection of ℓ partitions of U, $U^{(1)}, \ldots, U^{(\ell)}$, 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. Two canonical hierarchies are shown in Figure 2, in which $a, \ldots, j$ represent users and the rectangles define the partition blocks. Note that each level is a refinement of the one above.
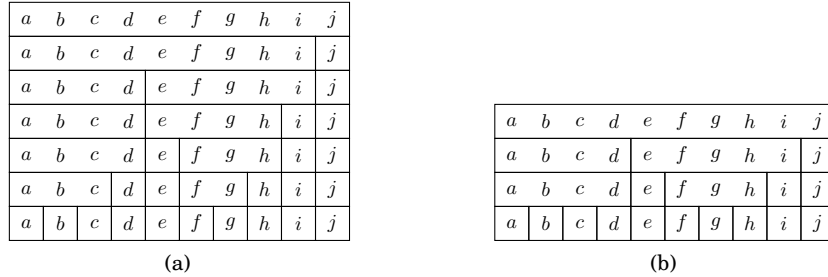

Fig. 2.   Two canonical organizational hierarchies

More formally, we have the following definition.

Definition 5.3. Let $\mathcal{H} = U^{(1)}, \ldots, U^{(\ell)}$, 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 $U^{(1)}, \ldots, U^{(\ell)}$ 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:
— 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 $\ell + 2$.

5.1. 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 $G_{\mathrm{man}} = (U, E_{\mathrm{man}})$ 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 $(u, v) \in E_{\mathrm{man}}$. We also assume that the out-degree of a non-leaf node is at least two.

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;
(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. 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(a). 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(a).

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
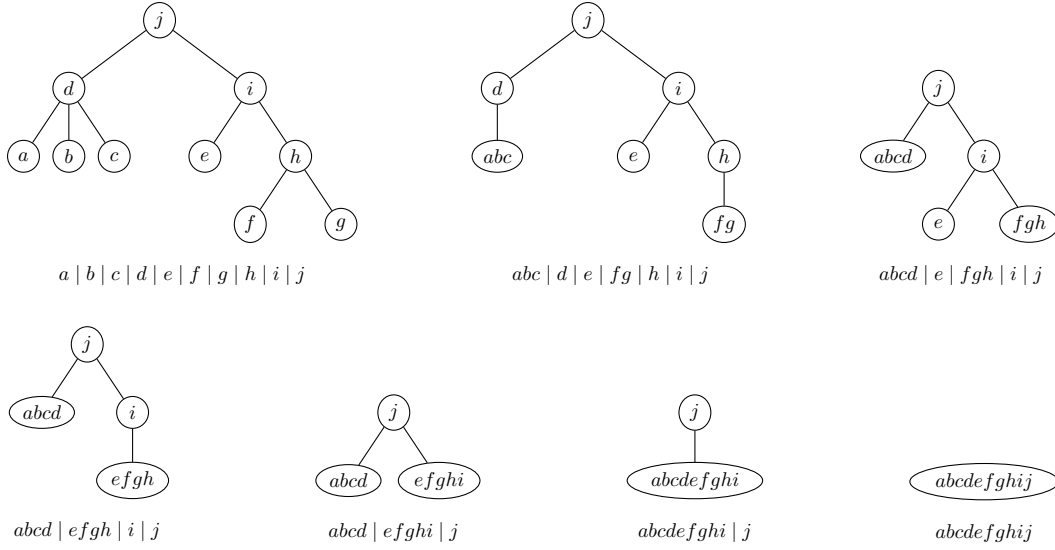
Fig. 3.   Building the blocks of an organizational hierarchy from a management tree

s_1 or s_2 because there would be 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 Moffett and Lupu [1999], 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 [Bertino et al. 1999; Crampton 2005; Wang and Li 2010], which have focused on constraints involving only $=$ and $\neq$. 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. At each iteration we could, for example, collapse the root node and all the leaf nodes into a single node. In doing so, we remove the distinction between the line manager of an organizational unit and the remaining members of the unit. If we adopt this approach for the management tree in Figure 3, we derive the organizational hierarchy shown in Figure 2(b). Clearly this construction results in fewer layers in the organizational hierarchy (equal to $p + 1$, where p is the length of the longest directed path in the management tree) and, therefore, supports fewer choices of workflow constraints.

Each method will give rise to different organizational hierarchies, some with more levels, some with fewer, with each hierarchy allowing for the specification of a different set of constraints. The method used to construct an organizational hierarchy will

usually depend on the workflow, the organization and the type of constraints that are required. An alternative approach to both those described above would be to "stratify" the management tree into levels and, working from the bottom level up, collapse all departments at a specific level into single nodes. Using the management tree in Figure 3, for example, the users f and g form the lowest level in the stratified tree and would be merged into a single unit first; this would be followed by the merging of users a, b and c and of e, f, g and h. The resulting canonical hierarchy will be rather similar to the one depicted in Figure 2(a), although the departments will form at different levels in the new hierarchy. The study of such hierarchies and the utility of the constraints that can be defined over them will be the subject of future work.

5.2. 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 $\text{WSP}_2(\sim, \not\sim)$ may be transformed into an instance of $\text{WSP}_2(=, \neq)$, which is known to be FPT. We prove in Theorem 5.4 that the problem remains in FPT for organizational hierarchies with ℓ levels (defined by ℓ equivalence relations). In fact, the results in Theorem 4.1 and Proposition 4.6 correspond to special cases of Theorem 5.4, in which the hierarchy has two levels. Figure 4 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 5.4 to solve WSP for more complex hierarchical structures.

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

Fig. 4.   Two-level hierarchies

Recall Wang and Li proved that WSP is not FPT, in general. One crucial factor in determining the complexity of WSP is the nature of the binary relations used to define entailment constraints. Informally, Wang and Li showed that for a particular choice of relational structure on the user set, WSP is an instance of INDEPENDENT SET, which is known to be W[1]-complete. Constraints based on equivalence relations, however, do not compromise the fixed parameter tractability of WSP because of the particular structure that is imposed on the user set—namely, a partition into no more than 2^k blocks.

Before proving the main result of this section, we consider canonical hierarchies with exactly three levels. There are several reasons for doing so:

— if we are given a non-trivial equivalence relation $\sim$ and we are interested in $\text{WSP}(=, \neq, \sim, \not\sim)$ then there are three levels in the organizational hierarchy;
— constraints containing $\sim$ and $\not\sim$ have useful applications for many types of authorization policies; and
— three-level hierarchies represent the "tipping point" at which WSP becomes hard, in the sense that no polynomial kernel exists (see Section 6).

There are several situations in which we may have a single non-trivial equivalence relation. Perhaps the most obvious one arises when a set of users is grouped into distinct departments or organizational units, as we have previously noted. Other possi-

bilities arise from a natural re-interpretation of the authorization relation $A \subseteq S \times U$: specifically, we define $u \sim_A u'$ if and only if u and u' are authorized for the same workflow steps. Then there are a maximum of 2^k equivalence classes (each associated with a particular subset of workflow steps). In a role-based view of authorization [Sandhu et al. 1996], a set of permissions (such as execution of workflow steps) defines a role. With this interpretation, a constraint of the form $(\sim, s_1, s_2)$ requires that s_1 and s_2 are performed by users that are assigned to the same role(s), with an analogous interpretation for $(\nsim, s_1, s_2)$.[10]

We may also consider an authorization policy that associates users and workflow steps with a security label, as in the Bell-LaPadula security model [Bell and LaPadula 1976]. More formally, let $(L, \leqslant)$ be a partially ordered set of security labels and $\lambda : U \cup S \to L$ a function that associates each user and step with a security label. Then a user is authorized to perform step s if and only if $(s, u) \in A$ and $\lambda(u) \geqslant \lambda(s)$. Clearly $\sim_\lambda$, where $u \sim_\lambda u'$ if and only if $\lambda(u) = \lambda(u')$ is an equivalence relation. The constraint $(\sim, s_1, s_2)$ requires that steps s_1 and s_2 be performed by users with the same security clearance. In short, there seem to be a number of situations in which the use of constraints defined by equivalence relations will be useful.

THEOREM 5.4. *Given a workflow* $(S, U, \leqslant, A, C)$ *and a canonical hierarchy with* ℓ *levels,* $\mathrm{WSP}_2(\sim_1, \nsim_1, \ldots, \sim_\ell, \nsim_\ell)$ *can be solved in time* $\widetilde{O}(3^k n(c + n))$*, where* n*,* k *and* c *are the numbers of users, steps and constraints, respectively.*

Theorem 4.1 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 $U^{(2)} = (U)$. To prove Theorem 5.4, we identify particular types of blocks in the hierarchy (those shaded in Figure 5) and solve multiple instances of WSP for each of those "significant" blocks. The results for significant blocks at a particular level are then used to solve instances of WSP for significant blocks at higher levels in the hierarchy.

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

Fig. 5.   The canonical hierarchy of Figure 3 with its significant blocks shaded

PROOF OF THEOREM 5.4.   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 $V \notin U^{(i+1)}$. 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 5 is $[3, 5]$, for example.

Each significant block V with level range $[a, b]$, $a > 1$, can be partitioned into blocks in level $(a - 1)$. We denote this set of blocks by $\Delta(V)$. Each significant block V with

---

[10]Of course, we could replace A with user- and permission-role assignment relations, but we could still derive the same equivalence classes.

level range $[1, b]$ comprises a single user (see Figure 5). 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, in which the leaf nodes are blocks with level range $[1, b]$ for some $b < \ell$.

Given an instance $\mathcal{I}$ of $\text{WSP}_2(\sim_1, \not\sim_1, \ldots, \sim_\ell, \not\sim_\ell)$, every subset F of S and every significant block V with children $\Delta(V)$ defines an instance of WSP in which:

— the set of steps is F;

— the set of users is $\Delta(V)$;

— the authorization relation A' is a subset of $F \times \Delta(V)$, where $(s, W) \in A'$ if and only if there exists a user in $v \in W$ such that $(s, v) \in A$;

— the set of constraints comprises those constraints in C of the form (ρ, S_1, S_2), where ρ is $\sim_i$ or $\not\sim_i$ with $a \leqslant i \leqslant b$.

We denote this derived instance of WSP by $\mathcal{I}_{F,V}$. Note that if V has level range $[1, b]$, then $\mathcal{I}_{F,V}$ asks whether a single user is authorized to perform all the steps in F without violating any constraints defined between levels 1 and b of the hierarchy. If V has level range $[a, b]$, with $a > 1$, then $\mathcal{I}_{F,V}$ is solved using the approach similar to that described in the proof of Theorem 3.4. When building the matrix, the entry indexed by $G \subseteq F$ and W is defined to be 0 if and only if $G \neq \emptyset$ is ineligible or $\mathcal{I}_{G,W}$ is a no-instance of WSP. Thus, a non-zero matrix entry indicates the steps in F could be assigned to the block W (meaning 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. Hence, we can solve $\mathcal{I}_{F,V}$ if we can solve $\mathcal{I}_{F,W}$ for all $W \in \Delta(V)$.

Note, finally, that U is a significant set and a solution for $\mathcal{I}_{S,U}$ is a solution for $\mathcal{I}$. Thus our algorithm for solving $\mathcal{I}$ solves $\mathcal{I}_{F,V}$ for all significant sets V with level range $[a, b]$ from $a = 1$ to $a = \ell$ and all subsets F of S.

We now consider the complexity of this algorithm. Consider the significant block V with m children. If $m = 0$ then $V = \{u\}$ for some $u \in U$ and solving $\mathcal{I}_{F,V}$ amounts to identifying whether F is an eligible set and whether u is authorized for all steps in F. For fixed V (with $m = 0$), solving $\mathcal{I}_{F,V}$ for all $F \subseteq S$ takes time $O(2^k c)$. There are exactly n significant sets, one per user, with no children. If $m > 0$ then the time taken to solve $\mathcal{I}_{F,V}$ is $\widetilde{O}(2^{|F|}(c + m^2))$, by Theorem 4.1. Hence the time taken to solve $\mathcal{I}_{F,V}$ for all $F \subseteq S$ (for fixed V) is $\widetilde{O}(3^k(c + m^2))$. 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 $|\mathcal{V}| \leqslant 2n - 1$), so there are at most $n - 1$ significant sets with 2 or more children.

The total time taken, therefore, is

$$O(2^k cn) + \sum_{V \in \mathcal{V}} \widetilde{O}(3^k(c + m_V^2) = \widetilde{O}(3^k cn) + \sum_{V \in \mathcal{V}} \widetilde{O}(3^k m_V^2),$$

where m_V denotes the number of children of V.

Now for some $b \geqslant 0$, we have

$$\sum_{V \in \mathcal{V}} \widetilde{O}(m_V^2) = \sum_{V \in \mathcal{V}} O((m_V \log^b m_V)^2) \leqslant \max_{V \in \mathcal{V}} \log^{2b} m_V \sum_{V \in \mathcal{V}} O(m_V^2) = O(n^2 \log^{2b} n) = \widetilde{O}(n^2).$$

Hence, we conclude that the total time taken to compute ϕ_V for all V is $\widetilde{O}(3^k cn + 3^k n^2)) = \widetilde{O}(3^k n(c + n))$. $\square$

Remark 5.5. The algorithm in the above proof can be optimized by computing a single matrix for each significant set V (with rows indexed by $\Delta(V)$ and columns indexed by subsets of S), which can be used to solve $\mathcal{I}_{F,V}$ for all $F \subseteq S$. This matrix can be built in time $O(cm2^k)$ and the solution to $\mathcal{I}_{F,V}$, for $F \subseteq S$, can be computed in time

$\widetilde{O}(2^{|F|}m^2)$. Hence, the optimized algorithm runs in time $\widetilde{O}(cm2^k + m^23^k)$, for fixed V, and in time $\widetilde{O}(cn2^k + n^23^k)$ overall.

THEOREM 5.6. *Let* $\sim_1, \ldots, \sim_\ell$ *define a canonical organizational hierarchy. Let* $W = (S, U, \leqslant, A, C \cup C_\sim \cup C_{\not\sim})$ *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 the satisfiability of* W *can be determined in time*

$$\widetilde{O}((c + 2c')n2^{k+c'} + n^23^{k+c'}),$$

where $c = |C| + |C_{\not\sim}|$ *and* $c' = |C_\sim|$. *Moreover,* $c' \leqslant 3^k$, *so* $\text{WSP}_3(\sim_1, \ldots, \sim_\ell, \not\sim_1, \ldots, \not\sim_\ell)$ *is FPT.*

The proof of this result can be found in the appendix.

6. KERNELIZATION

Formally, a *parameterized problem* P can be represented as a relation $P \subseteq \Sigma^* \times \mathbb{N}$ over a finite alphabet Σ. The second component is call the *parameter* of the problem. In particular, WSP is a parameterized problem with parameter k, the number of steps. We denote the size of a problem instance (x, k) by $|x| + k$. In this section, we are interested in transforming an instance of WSP into a new instance of WSP whose size is dependent only on k. This type of transformation is captured in the following definition.

Definition 6.1. Given a parameterized problem P, a *kernelization of* P is an algorithm that maps an instance (x, k) to an instance (x', k') in time polynomial in $|x| + k$ such that (i) $(x, k) \in P$ if and only if $(x', k') \in P$, and (ii) $k' + |x'| \leqslant g(k)$ for some function g; (x', k') is the *kernel* and g is the *size* of the kernel.

Note that a kernelization provides a form of preprocessing aimed at compressing the given instance of the problem. The compressed instance can be solved using *any* suitable algorithm (such as a SAT solver), not necessarily by an FPT algorithm. It is well-known and easy to prove that a decidable parameterized problem is FPT if and only if it has a kernel [Flum and Grohe 2006]. If $g(k) = k^{O(1)}$, then we say (x', k') is a *polynomial-size* kernel.

Polynomial-size kernels are particularly useful in practice as they often allow us to reduce the size of the input of the problem under consideration to an equivalent problem with an input of significantly smaller size. This preprocessing often allows us to solve the original problem more quickly. Unfortunately, many fixed-parameter tractable problems have no polynomial-size kernels (unless coNP $\subseteq$ NP/poly, which is highly unlikely [Bodlaender et al. 2009; Bodlaender et al. 2011a; Bodlaender et al. 2011b; Dom et al. 2009]).

In order to illustrate the benefits of kernelization, we first state and prove three simple results, the first two of which extend a result of Fellows et al. [2011]. We then show that $\text{WSP}_1(=, \neq)$ has a kernel with at most k users.

PROPOSITION 6.2. $\text{WSP}(\neq)$ *has a kernel with at most* $k(k - 1)$ *users. Moreover, a kernel with at most* $k(k - 1)$ *users exists if we extend the set of constraints to include counting constraints of the form* $(1, t, S')$.

PROOF. Let $W = (S, U, \leqslant, A, C)$ be a workflow in which all constraints have the form $(\neq, S_1, S_2)$. Let S_{easy} be the set of steps such that each step has at least k authorized users and let S_{hard} be $S \setminus S_{\text{easy}}$. Now consider the workflow $W_{\text{hard}} = (S_{\text{hard}}, U_{\text{hard}}, \leqslant, A_{\text{hard}}, C_{\text{hard}})$, where $u \in U_{\text{hard}}$ if and only if u is authorized for at least one step in

S_{hard}, $A_{\text{hard}} = (S_{\text{hard}} \times U) \cap A$, and $(\neq, S_1, S_2) \in C_{\text{hard}}$ if and only if $(\neq, S_1, S_2) \in C$ and $S_1, S_2 \subseteq S_{\text{hard}}$. A counting constraint of the form $(1, t_r, S')$ is replaced by the counting constraint $(1, t_r, S' \setminus S_{\text{easy}})$.

We now solve the WSP instance defined by W_{hard} and show that this allows us to compute a solution for W. If W_{hard} is a no instance, then W cannot be satisfiable either (since $C_{\text{hard}} \subseteq C$). Conversely, if it is a yes instance, then there exists a plan $\pi_{\text{hard}} : S_{\text{hard}} \to U_{\text{hard}}$. Moreover, we can extend π_{hard} to a plan $\pi : S \to U$, so W is satisfiable. Specifically, we allocate a different user from $U \setminus \pi_{\text{hard}}(S_{\text{hard}})$ to each step in $s \in S_{\text{easy}}$ (which is possible since there are at least k users authorized to perform s and only k steps in total) and define $\pi(s) = \pi_{\text{hard}}(s)$ for all $s \in S_{\text{hard}}$. Clearly, π does not violate any constraint of the form $(\neq, S_1, S_2)$ or $(1, t, S')$.[11]

In other words, we can solve WSP for W by solving WSP for W_{hard}, which has no more than k steps and each step has fewer than k authorized users. Hence, there can be no more than $k(k-1)$ authorized users in W_{hard}. □

COROLLARY 6.3. $\text{WSP}_1(\neq)$ *can be solved in time* $\widetilde{O}(2^k)$.

PROOF. The result follows immediately from Theorem 4.1, the fact that there can be no more than $O(k^2)$ Type 1 constraints, and the proposition above. □

PROPOSITION 6.4. $\text{WSP}_1(\neq, =)$ *has a kernel with at most* $k(k-1)$ *users.*

PROOF. The basic idea is to merge all steps that are related by constraints of the form $(=, s_1, s_2)$ for $s_1, s_2 \in S$. More formally, consider an instance $\mathcal{I}$ of $\text{WSP}_1(=, \neq)$, given by a workflow $(S, U, \leqslant, A, C)$.

(1) Construct a graph H with vertices S, in which $s', s'' \in S$ are adjacent if C includes a constraint $(=, s', s'')$.
(2) If there is a connected component of H that contains both s' and s'' and C contains a constraint $(\neq, s', s'')$ then $\mathcal{I}$ is unsatisfiable, so we may assume there is no such connected component.
(3) For each connected component T of H,
    (a) replace all steps of T in S by a "superstep" t;
    (b) for each such superstep t, authorize user u for t if and only if u was authorized (by A) for all steps in t
    (c) for each such superstep t, merge all constraints for steps in t.

Clearly, we now have an instance of $\text{WSP}_1(\neq)$, perhaps with fewer steps and a modified authorization relation, that is satisfiable if and only if $\mathcal{I}$ is satisfiable. The result now follows by Proposition 6.2.

The reduction can be performed in time $O(kc + kn)$, where c is the number of constraints: step (1) takes time $O(k + c)$; step (3) performs at most k merges; each merge takes $O(k + c + n)$ time (since we need to merge vertices, and update constraints and the authorization relation for the new vertex set);[12] finally, if $k \leqslant c$ we have $O(k(k + c + n) = O(k(c + n))$, and if $c \leqslant k$ then we perform no more than c merges in time $O(c(k + c + n)) = O(ck + cn) = O(ck + kn)$. □

THEOREM 6.5. $\text{WSP}_1(=, \neq)$ *admits a kernel with at most* k *users.*

PROOF. We first use the WSP_1 constraint reduction method from the proof of Proposition 6.4 to eliminate all constraints of the form $(=, s', s'')$, leaving an instance $\mathcal{I}$ of

---

[11]Note that this is not true for counting constraints of the form (t_ℓ, t_r, S') when $t_\ell > 1$.

[12]We can check step (2) when we merge constraints in step 3(c).

WSP$_1(\neq)$. We now construct a bipartite graph $G = (U, S; A)$, where $A \subseteq S \times U$ is the authorization relation. We may assume that $|U| \geqslant |S| = k$.

Let $V = U \cup S$. Using the well-known Hopcroft-Karp algorithm, we can find a maximum matching M in G in time $O(\sqrt{|V|}|A|)$.[13] If M covers every vertex of S, then $\mathcal{I}$ is satisfiable and our kernel is the subgraph of G induced by all vertices covered by M. (Since there is at most one edge in M for each vertex in S and at most one edge for each vertex in U, there are exactly k users covered and we have a kernel containing k users.)

If M does not cover every vertex of S then we define $R_{G,M}$ to be the set of vertices of G which can be reached from some uncovered vertex in S by an M-alternating path.[14] Then a result of Szeider [2004, Lemma 3] asserts that we can compute $R_{G,M}$ in time $O(|U| + |S| + |A|)$. We write $R_{G,M}$ in the form $U' \cup S'$ for some $U' \subseteq U$ and $S' \subseteq S$. The set $U' \cup S'$ has the following properties [Szeider 2004, Lemma 3]:
P1. All vertices of $S \setminus S'$ are covered by M;
P2. There is no edge in G from $U \setminus U'$ to S' and no edge of M joins vertices in U' with vertices in $S \setminus S'$;
P3. In the subgraph G induced by $U' \cup S'$, vertices of a set $U'' \subseteq U'$ have at least $|U''| + 1$ neighbors in S'.
A bipartite graph G, a maximum matching M in G (indicated by the thicker lines), and the sets U' and S' are shown in Figure 6; the figure is based on one used by Szeider [2004].
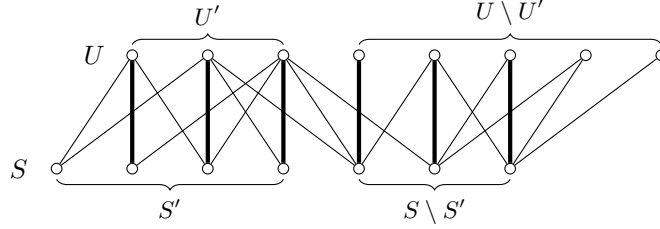

Fig. 6.   Constructing a kernel for WSP using a maximum matching

Hence, we can assign users to all steps that are not in S' (using M) and we will not violate any separation-of-duty constraints by doing so. Moreover, property (P2) means that allocating users in U' to steps in S' will not violate any separation-of-duty constraints. In other words, we have reduced the problem instance to finding a solution to a smaller instance (the kernel) in which the set of users is U', the set of steps is S', and $|U'| < |S'| \leqslant k$. □

The authorization relation $A \subseteq S \times U$ defines the bipartite graph used to construct the matching. The computation of a maximum matching in time $O(|A| \cdot \sqrt{n + k}) = O(nk\sqrt{n + k})$ enables us to compute a partial plan π, where an edge in the matching corresponds to a step s and a user $u = \pi(s)$. If the maximum matching has cardinality k, then we are done. Otherwise, we solve WSP for the kernel.

When the cardinality of A is high (so the computation of the maximum matching is relatively slow), many users are authorized for many steps. In this case, therefore, the

---

[13]A matching in a bipartite graph is a set of edges that are pairwise non-adjacent. A maximum matching contains the largest possible number of edges.
[14]An M-alternating path has the property that for any pair of successive edges one belongs to M and the other does not.

observation that only those steps for which fewer than k users are authorized need to be considered may mean that it is easy to decide whether the instance is satisfiable.

We now state some negative results, negative in the sense that they assert that certain instances of WSP do not have polynomial-size kernels. The proofs of these results can be found in the appendix.

THEOREM 6.6. $WSP_2(=)$ *does not admit a kernel with a polynomial number of users unless coNP* $\subseteq$ *NP/poly.*

THEOREM 6.7. *WSP with counting constraints of the type* $(2, t, S')$ *does not admit a kernel with a polynomial number of users unless coNP* $\subseteq$ *NP/poly.*

The above results tells us that there may be little to be gained from preprocessing an instance of $WSP_2(=)$ or an instance that contains arbitrary counting constraints, and we may simply apply the techniques described in Section 4. Our final result in this section proves that the existence of a polynomial kernel is unlikely when we consider WSP for canonical organizational hierarchies, even when we restrict attention to Type 1 constraints and hierarchies with only three levels.

THEOREM 6.8. *The problem* $WSP_1(=, \neq, \sim, \nsim)$*, where* $\sim$ *is an equivalence relation defined on* U*, does not have a polynomial kernel, unless NP* $\subseteq$ *coNP/poly.*

7. CONCLUDING REMARKS

In general terms, the results reported in this paper provide a much improved understanding of the fixed parameter tractability of the workflow satisfiability problem. In particular, we have developed a technique—the reduction of WSP to MAX WEIGHTED PARTITION—that guarantees an instance of WSP is FPT, provided all constraints satisfy two simple criteria. This enables the designer of workflow systems to determine whether the satisfiability of a workflow specification is FPT by examining the constraints defined in the specification. Our results in this paper achieve several specific things.

— First, the use of the MAX WEIGHTED PARTITION problem to solve WSP allows us to develop a fixed-parameter algorithm for which the worst-case run-time is significantly better than known algorithms.
— Second, this algorithm can be used to solve more general constraints—counting constraints, Type 3 entailment constraints and constraints based on equivalence relations—than was possible with existing methods. In short, we have extended the classes of workflow specifications for which the satisfiability problem is known to be FPT.
— Third, we have established the circumstances under which an instance of WSP has a polynomial kernel. As well as providing the first results of this type for WSP, kernelization is of enormous practical value. The computation of a maximum matching in time $O(nk\sqrt{n+k})$ is an extremely useful technique for deriving a (partial) plan for an instance of WSP. Moreover, the reduction in the size of the problem instance when the maximum matching generates a partial plan will significantly reduce the complexity of solving instances of $WSP_1(=, \neq)$.
— Finally, we have significantly extended our understanding of those instances of WSP that are FPT. Specifically, WSP is FPT for any workflow specification that only includes constraints that are regular and for which (in)eligibility can be determined in time polynomial in the number of steps. In particular, we have established that WSP problems which include constraints based on counting constraints and on user equivalence classes—enabling us to model organizational structures and business rules defined in terms of those structures—are still FPT.

In short, we believe our results represent a significant step forward in our understanding of the complexity of WSP and provide the blueprints for algorithms that can find efficient solutions for many practical instances of WSP.

7.1. 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. [1999]. 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 [Crampton 2005]. 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 $WSP_2(=, \neq)$ [Wang and Li 2010]. Moreover, they established that WSP_2 is W[1]-hard, in general.

Recent work by Basin et al. [2011] 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. 2011] studied $WSP_1(\neq)$ 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 4.1). Moreover, our work establishes that it is unlikely that our bound can be significantly improved (Theorem 4.2). We extend the type of constraints that can be defined by introducing counting constraints and Type 3 entailment constraints, and we have shown that WSP remains fixed-parameter tractable (Theorems 3.6 and 5.6).

Most recently, we showed how WSP for entailment constraints could be reduced to MAX WEIGHTED PARTITION for particular constraint relations. In this paper, we have extended our approach to include any form of constraint that is regular and for which eligibility can be determined in time polynomial in the number of steps. This represents a significant advance as it means we need only test whether a constraint is regular and devise an efficient eligibility test to deploy our techniques for solving WSP.

7.2. 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. In particular, we hope to identify which security requirements can be encoded using constraints that satisfy the criteria identified in Theorem 3.4. A very natural relationship between users is that of seniority: we would like to establish whether the inclusion of constraints based on this binary relation affects the fixed-parameter tractability of WSP.

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) [van der Aalst et al. 2003]. 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 iterated sub-workflows. We will consider the fixed-parameter tractability of EPEP, and WSP for richer workflow patterns, in our future work.

Wang and Li also introduced the notion of workflow *resiliency*. The *static t-resiliency checking problem (SRCP)* asks whether a workflow specification remains satisfiable if some subset of t users is absent. Clearly SRCP is NP-hard as the case $t = 0$ corresponds to WSP. Evidently, SRCP can be resolved by considering the $\binom{n}{t}$ instances of WSP that can arise when t users are absent. Hence, SRCP is in coNP$^{\text{NP}}$ [Wang and Li 2010, Theorem 13]. The problems of deciding whether a workflow has *dynamic* or *decremental* t-resiliency are PSPACE-complete [Wang and Li 2010, Theorems 14–15]. Basin et al. [2012] study a related problem called the *optimal workflow-aware authorization administration problem*, which determines whether it is possible to modify the authorization relation, subject to some bound on the "cost" of the changes, when the workflow is unsatisfiable. It will be interesting, therefore, to explore whether we can better understand the parameterized complexity of these kinds of problems.

A. PROOFS OF THEOREMS

In this appendix, we provide proofs of Theorems 4.2, 6.6, 6.8 and 5.6. Before proving Theorem 4.2, we define two problems related to 3-SAT and state two preparatory lemmas.

---

c-LINEAR-3-SAT
  Input: A 3-CNF formula ϕ with m clauses, and n variables such that $m \leqslant cn$,
       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.* [Impagliazzo et al. 2001] (see also [Crowston et al. 2012]).

LEMMA A.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 A.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 A.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

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

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 4.2. 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 A.2. □

Before proving Theorem 6.6, we introduce a definition and result due to Bodlaender et al. [2011b].

Definition A.3. Let P and Q be parameterized problems. We say a polynomial time computable function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ is a *polynomial parameter transformation* from P to Q if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for any $(x, k) \in \Sigma^* \times \mathbb{N}$, $(x, k) \in P$ if and only if $f(x, k) = (x', k') \in Q$, and $k' \leq p(k)$.

LEMMA A.4. [Bodlaender et al. 2011b, Theorem 3] *Let P and Q be parameterized problems, and suppose that P^c and Q^c are the derived classical problems (where we disregard the parameter). Suppose that P^c is NP-complete, and $Q^c \in NP$. Suppose that f is a polynomial parameter transformation from P to Q. Then, if Q has a polynomial-size kernel, then P has a polynomial-size kernel.*

PROOF OF THEOREM 6.6. We may formulate the HITTING SET problem as a problem for bipartite graphs. We are given a bipartite graph with with partite sets $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_m\}$ and edge set E. We are to decide whether there is a subset H of U with at most k vertices such that each $v \in V$ has a neighbor in H.

We say that two problems are *equivalent* if every yes instance of one corresponds to a yes instance of the other. Wang and Li [2010, Lemma 4] proved that HITTING SET is equivalent to the following subproblem Π of $\mathrm{WSP}_2(=)$. We have U as the set of users, $V \cup S$ as the set of $k' = m+k$ steps, every user u_j is authorized to perform any step from S and every step v_i such that $u_j v_i \in E$, and $(=, v_i, S)$, $i \in [m]$, is the set of constraints of Type 2.

Observe that the above construction gives a polynomial parameter transformation from HITTING SET parameterized by $m+k$ to $\mathrm{WSP}_2(=)$. Dom et al. [2009] proved that HITTING SET parameterized by $m+k$ does not admit a polynomial-size kernel unless coNP $\subseteq$ NP/poly. Now we are done by Lemma A.4. □

PROOF OF THEOREM 6.7. We will use the polynomial parameter transformation from HITTING SET parameterized by $m+k$ to a subproblem Π of WSP described in the proof of Theorem 6.6. We obtain a subproblem Π^* of WSP with counting constraints of the type $(2, t, S')$ from Π by keeping the same set U of users and the same set $V \cup S$ of steps, but by replacing the constraints of Π with $(2, k+1, S \cup \{v_i\})$, $i \in [m]$.

We now prove that Π and Π^* are equivalent, from which the result follows by Theorem 6.6.

Let π^* be a valid plan for Π^* and let π be obtained from π^* by restricting it to $V \cup S$. Observe that if a constraint $(2, k+1, S \cup \{v_i\})$ is satisfied by π^*, then $(=, v_i, S)$ is satisfied by π. Thus, π is a valid plan for Π.

Let π be a valid plan for Π and let π^* be obtained from π by reassigning to $\pi(v_1)$ every step s in S such that the user $\pi(s)$ is assigned to perform just one step in $V \cup S$. Observe that if $(=, v_i, S)$ is satisfied by π, then $(2, k+1, S \cup \{v_i\})$ is satisfied by π^*. Thus, π^* is a valid plan for Π^*. □

The following two definitions and Theorem A.7 are due to Bodlaender et al. [2011a].

Definition A.5 (*Polynomial equivalence relation*). An equivalence relation $\mathcal{R}$ on Σ^* is called a *polynomial equivalence relation* if the following two conditions hold:
— There is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether x and y belong to the same equivalence class in $(|x| + |y|)^{O(1)}$ time.
— For any finite set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of S into at most $(\max_{x \in S} |x|)^{O(1)}$ equivalence classes.

Definition A.6 (*Cross-composition*). Let $L \subseteq \Sigma^*$ be a problem and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that L *cross-composes* into Q if there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm which, given t strings $x_1, \ldots, x_t$ belonging to the same equivalence class of $\mathcal{R}$, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that:
— $(x^*, k^*) \in Q$ if and only if $x_i \in L$ for some $1 \leqslant i \leqslant t$.
— k^* is bounded by a polynomial in $\max_{i=1}^{t} |x_i| + \log t$.

THEOREM A.7. *If some problem L is NP-hard under Karp reductions and L cross-composes into the parameterized problem Q then there is no polynomial kernel for Q unless NP $\subseteq$ coNP/poly.*

PROOF OF THEOREM 6.8. We may treat $\mathrm{WSP}_1(=, \neq, \sim, \nsim)$ as an instance of $\mathrm{WSP}_1(\sim_1, \nsim_1, \sim_2, \nsim_2, \sim_3, \nsim_3)$ for a canonical hierarchy with three levels, where $\sim_1$ and $\nsim_1$ correspond to $=$ and $\neq$ respectively.

We will use Theorem A.7 to show the result, hence we need an NP-hard problem L which cross-composes into $\mathrm{WSP}_1(\sim_1, \not\sim_1, \sim_2, \not\sim_2, \sim_3, \not\sim_3)$. For this purpose we will use the problem 3-COLORING. An instance of 3-COLORING is a graph G in which we want to decide if it can be 3-colored. We say that two graphs G_1 and G_2 are equivalent if $|V(G_1)| = |V(G_2)|$. It is not difficult to see that this defines a polynomial equivalence relation on 3-COLORING (see Definition A.5).

Consider now t instances of 3-COLORING, $G_1, G_2, \ldots, G_t$. Let

$$k = |V(G_1)| = |V(G_2)| = \cdots = |V(C_t)| \quad \text{and} \quad V(G_i) = \{x_1^i, x_2^i, \ldots, x_k^i\}.$$

We now construct an instance of $\mathrm{WSP}_1(\sim_1, \not\sim_1, \sim_2, \not\sim_2, \sim_3, \not\sim_3)$ with steps S and users U defined as follows.

$$S = (\cup_{i=1}^k V_i) \cup_{1 \leqslant i < j \leqslant k} \{e_{i,j}, e_{i,j}'\} \quad \text{where } V_i = \{v_1^i, v_2^i\};$$

$$U = \cup_{i=1}^t U_i \qquad\qquad\qquad\qquad \text{where } U_i = \{c_1^i, c_2^i, c_3^i, \alpha^i\}.$$

Observe that $|S| = k + k^2$ is bounded by a polynomial in the maximum size of G_i, $i \in [t]$.

We now define the hierarchy $\mathcal{H} = U^{(1)}, U^{(2)}, U^{(3)}$, where $U^{(1)}$ is the partition of S containing all singletons, $U^{(2)}$ is the partition $U_1, U_2, \ldots, U_t$ and $U^{(3)}$ is the partition containing just one set S. We now define the constraints C as follows.

$$
\begin{aligned}
C = &\{(\not\sim_1, v_1^i, v_2^i) \mid i \in [k]\} \cup \\
&\{(\not\sim_1, v_1^i, e_{i,j}), (\not\sim_1, v_2^i, e_{i,j}), (\not\sim_1, e_{i,j}, e_{i,j}'), (\not\sim_1, v_1^j, e_{i,j}'), (\not\sim_1, v_2^j, e_{i,j}') \mid 1 \leqslant i < j \leqslant k\} \\
&\cup \{(\sim_2, s_1, s_2) \mid s_1, s_2 \in S\}
\end{aligned}
$$

We now let all users, except $\alpha^1, \alpha^2, \ldots, \alpha^t$, be authorized for all steps. Furthermore if $x_i^a x_j^a \notin E(G_a)$, where $1 \leqslant i < j \leqslant k$ then authorize α^a for $e_{i,j}$.

Claim A. The created instance has a valid plan if and only if one of the graphs, $G_1, G_2, \ldots, G_t$ are 3-colorable. The result now follows by Theorem A.7.

Proof of Claim A. Assume that the created instance has a valid plan. The constraints $\{(\sim_2, s_1, s_2) \mid s_1, s_2 \in S\}$ imply that all users used in the plan belong to exactly one block in $U^{(2)}$, say U_r. Let $\gamma_j \in \{1, 2, 3\}$ be defined such that the users assigned to v_1^j and v_2^j are $\{c_1^r, c_2^r, c_3^r\} \setminus \{c_{\gamma_j}^r\}$, which is possible as $(\not\sim_1, v_1^j, v_2^j) \in C$ and $\{c_1^r, c_2^r, c_3^r\}$ are the only users from U_r authorized for $\{v_1^j, v_2^j\}$. If $x_i^r x_j^r \in E(G_r)$ then $e_{i,j}$ must be assigned user $c_{\gamma_i}^r$ and $e_{i,j}'$ must be assigned user $c_{\gamma_j}^r$, which implies that $\gamma_i \neq \gamma_j$, by the given constraints. Therefore $\gamma_1, \gamma_2, \ldots, \gamma_k$ is a 3-coloring of G_r. This shows one direction of Claim A.

Now assume we have a 3-coloring $\gamma_1, \gamma_2, \ldots, \gamma_k$ of G_r, for some $r \in [t]$. Assign users $\{c_1^r, c_2^r, c_3^r\} \setminus \{c_{\gamma_j}\}$ to the steps v_1^j and v_2^j and for all $e_{i,j}$ assign user $c_{\gamma_i}^r$ if $x_i^r x_j^r \in E(G_r)$ or user α_r if $x_i^r x_j^r \notin E(G_r)$. Finally assign user $c_{\gamma_j}^r$ to all steps $e_{i,j}'$. Note that the given assignment of users satisfies all constraints, which completes the proof of the claim. □

PROOF OF THEOREM 5.6. The result follows from a very similar argument to that used in the proof of Theorem 5.4. 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 $C_\sim$. Nevertheless, we can rewrite the set of constraints in $C_\sim$ as Type 2 constraints, at the cost of introducing additional workflow steps (as we did in the proof of Theorem 4.3). This requires the replacement of c' Type 3 constraints by $2c'$ Type 2 constraints and the creation of c' new steps. Finally, we solve the resulting instance of WSP_2 for a workflow with n users, $k + c'$ steps and $c + 2c'$ constraints, which has complexity $\widetilde{O}((c + 2c')n2^{k+c'} + n^2 3^{k+c'})$, by Theorem 5.4 and Remark 5.5.

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 $\sum_{j=1}^{k} \binom{k}{j} 2^{k-j} = 3^k$. $\quad\square$

REFERENCES

American National Standards Institute 2004. *ANSI INCITS 359-2004 for Role Based Access Control*. American National Standards Institute.

ARMANDO, A., GIUNCHIGLIA, E., AND PONTA, S. E. 2009. Formal specification and automatic analysis of business processes under authorization constraints: An action-based approach. In *TrustBus*, S. Fischer-Hübner, C. Lambrinoudakis, and G. Pernul, Eds. Lecture Notes in Computer Science Series, vol. 5695. Springer, 63–72.

BASIN, D. A., BURRI, S. J., AND KARJOTH, G. 2011. Obstruction-free authorization enforcement: Aligning security with business objectives. In *CSF*. IEEE Computer Society, 99–113.

BASIN, D. A., BURRI, S. J., AND KARJOTH, G. 2012. Optimal workflow-aware authorizations. In *SACMAT*, V. Atluri, J. Vaidya, A. Kern, and M. Kantarcioglu, Eds. ACM, 93–102.

BELL, D. AND LAPADULA, L. 1976. Secure computer systems: Unified exposition and Multics interpretation. Tech. Rep. MTR-2997, Mitre Corporation, Bedford, Massachusetts.

BERTINO, E., BONATTI, P. A., AND FERRARI, E. 2001. TRBAC: A temporal role-based access control model. *ACM Trans. Inf. Syst. Secur. 4,* 3, 191–233.

BERTINO, E., FERRARI, E., AND ATLURI, V. 1999. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur. 2,* 1, 65–104.

BJÖRKLUND, A., HUSFELDT, T., AND KOIVISTO, M. 2009. Set partitioning via inclusion-exclusion. *SIAM J. Comput. 39,* 2, 546–563.

BODLAENDER, H. L., DOWNEY, R. G., FELLOWS, M. R., AND HERMELIN, D. 2009. On problems without polynomial kernels. *J. Comput. Syst. Sci. 75,* 8, 423–434.

BODLAENDER, H. L., JANSEN, B. M. P., AND KRATSCH, S. 2011a. Cross-composition: A new technique for kernelization lower bounds. In *STACS*, T. Schwentick and C. Dürr, Eds. LIPIcs Series, vol. 9. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 165–176.

BODLAENDER, H. L., THOMASSÉ, S., AND YEO, A. 2011b. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci. 412,* 35, 4570–4578.

BREWER, D. F. C. AND NASH, M. J. 1989. The Chinese Wall security policy. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 206–214.

CRAMPTON, J. 2005. A reference monitor for workflow systems with constrained task execution. In *SACMAT*, E. Ferrari and G.-J. Ahn, Eds. ACM, 38–47.

CROWSTON, R., GUTIN, G., JONES, M., RAMAN, V., AND SAURABH, S. 2012. Parameterized complexity of MaxSat above average. In *LATIN*, D. Fernández-Baca, Ed. Lecture Notes in Computer Science Series, vol. 7256. Springer, 184–194.

DOM, M., LOKSHTANOV, D., AND SAURABH, S. 2009. Incompressibility through colors and IDs. In *ICALP (1)*, S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. E. Nikoletseas, and W. Thomas, Eds. Lecture Notes in Computer Science Series, vol. 5555. Springer, 378–389.

DOWNEY, R. G. AND FELLOWS, M. R. 1999. *Parameterized Complexity*. Springer Verlag.

FELLOWS, M. R., FRIEDRICH, T., HERMELIN, D., NARODYTSKA, N., AND ROSAMOND, F. A. 2011. Constraint satisfaction problems: Convexity makes AllDifferent constraints tractable. In *IJCAI*, T. Walsh, Ed. IJCAI/AAAI, 522–527.

FLUM, J. AND GROHE, M. 2006. *Parameterized Complexity Theory*. Springer Verlag.

IMPAGLIAZZO, R., PATURI, R., AND ZANE, F. 2001. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci. 63,* 4, 512–530.

JOSHI, J., BERTINO, E., LATIF, U., AND GHAFOOR, A. 2005. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng. 17,* 1, 4–23.

KAUFMAN, T., KRIVELEVICH, M., AND RON, D. 2004. Tight bounds for testing bipartiteness in general graphs. *SIAM J. Comput. 33,* 6, 1441–1483.

KOHLER, M. AND SCHAAD, A. 2008. Proactive access control for business process-driven environments. In *ACSAC*. IEEE Computer Society, 153–162.

LI, N., TRIPUNITARA, M. V., AND BIZRI, Z. 2007. On mutually exclusive roles and separation-of-duty. *ACM Trans. Inf. Syst. Secur. 10,* 2.

MOFFETT, J. D. AND LUPU, E. 1999. The uses of role hierarchies in access control. In *ACM Workshop on Role-Based Access Control*. ACM, 153–160.

SANDHU, R. S., COYNE, E. J., FEINSTEIN, H. L., AND YOUMAN, C. E. 1996. Role-based access control models. *IEEE Computer 29,* 2, 38–47.

SIMON, R. T. AND ZURKO, M. E. 1997. Separation of duty in role-based environments. In *CSFW*. IEEE Computer Society, 183–194.

SZEIDER, S. 2004. Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. *J. Comput. Syst. Sci. 69,* 4, 656–674.

TSANG, E. P. K. 1993. *Foundations of Constraint Satisfaction*. Academic Press. Available on-line at http://en.scientificcommons.org/43256499.

VAN DER AALST, W. M. P., TER HOFSTEDE, A. H. M., KIEPUSZEWSKI, B., AND BARROS, A. P. 2003. Workflow patterns. *Distributed and Parallel Databases 14,* 1, 5–51.

WANG, Q. AND LI, N. 2010. Satisfiability and resiliency in workflow authorization systems. *ACM Trans. Inf. Syst. Secur. 13,* 4, 40.