Artifact-Centric Workflow Dominance

Diego Calvanese[1], Giuseppe De Giacomo[2], Richard Hull[3], and Jianwen Su[4]

[1] KRDB Research Centre
Free University of Bozen-Bolzano
I-39100 Bolzano, Italy
calvanese@inf.unibz.it
[2] Dipartimento di Informatica e Sistemistica
Sapienza Università di Roma
I-00185 Roma, Italy
degiacomo@dis.uniroma1.it
[3] IBM T.J. Watson Research Center
Yorktown Heights, NY, U.S.A.
hull@us.ibm.com
[4] Department of Computer Science
University of California at Santa Barbara
Santa Barbara, CA, U.S.A.
su@cs.ucsb.edu

Abstract. In this paper we initiate a study on comparing artifact-centric workflow schemas, in terms of the ability of one schema to emulate the possible behaviors of another schema. Artifact-centric workflows are centered around "business artifacts", which contain both a *data schema*, which can hold all of the data about a key business entity as it passes through a workflow, along with a *lifecycle schema*, which specifies the possible ways that the entity can evolve through the workflow. In this paper, the data schemas for artifact types are finite sets of attribute-value pairs, and the lifecycle schemas are specified as sets of condition-action rules, where the condition is evaluated against the current snapshot of the artifact, and where the actions are external *services* (or "tasks"), which read a subset of the attributes of an artifact, which write onto a subset of the attributes, and which are performed by an entity outside of the workflow system (often a human). The services are also characterized by pre- and post-conditions, in the spirit of semantic web services. To compare artifact-centric workflows, we introduce the notion of "dominance", which intuitively captures the fact that all executions of a workflow can be emulated by a second workflow. (In the current paper, the emulation is focused only on the starting and ending snapshots of the possible enactments of the two workflows.) In fact, dominance is a parametric notion that depends on the characterization of the policies that govern the execution of the services invoked by the workflows. In this paper, we study in detail the case of "absolute dominance", in which this policy places no constraints on the possible service executions. We provide decidability and complexity results for bounded and unbounded workflow executions in the cases where the values in an artifact range over an infinite structure, such as the integers, the rationals, or the reals, possibly with order, addition, or multiplication.

L. Baresi, C.-H. Chi, and J. Suzuki (Eds.): ICSOC-ServiceWave 2009, LNCS 5900, pp. 130–143, 2009.

1 Introduction

The importance of automation of workflow and business processes continues to increase, with a world economy moving towards increased globalization and the drive for more efficiency. A fundamental problem in workflow management is to understand when one workflow (schema) can emulate another one. At a practical level, this is important for workflow evolution and workflow integration, where one might need to verify that the new workflow can faithfully emulate the old workflow(s). Emulation has been studied in considerable depth in the form of simulation for process algebras, which can be viewed as an abstraction of process-centric workflow models. In the past several years a data-centric approach to modeling workflows has emerged, in which both data and process are tightly coupled in the basic building blocks of workflows. One class of data-centric workflow models is centered around "business artifacts" [12,16], which are augmented data records that correspond to key business-relevant entities, their lifecycles, and how/when services (a.k.a. tasks) are invoked on them. This "artifact-centric" approach provides a simple and robust structure for workflow, and has been demonstrated in practice to permit efficiencies in business transformation [2,3]. In the artifact-centric approach obviously the process is of interest, but differently from process-centric workflows the data play a key role.

This paper provides a first investigation into workflow emulation in the context of artifact-centric workflows. In particular, we develop a basic framework for characterizing when one artifact-centric workflow "dominates" another one, and then provide decidability and complexity results for bounded and unbounded workflow executions for a particular kind of dominance, called "absolute dominance."

In our formal model, which follows the spirit of [4,7], the artifact "data schema" is a set of attribute-value pairs, which is used to hold relevant information about the artifact as it passes through the workflow. The values range over an infinite structure, such as the integers, the rationals, or the reals, possibly with order, addition, or multiplication. The "lifecycle schema", which is used to specify the possible ways that the artifact can pass through the workflow, is specified as a set of condition-action rules, where the condition is evaluated against the current snapshot of the artifact, and where the actions are *services* (a.k.a. "tasks"), which read a subset of the attributes of an artifact, which write onto a subset of the attributes, and which are performed by an entity outside of the workflow system (often a human). Similar to the context of semantic web services [11], the behaviors of the services used here are characterized using pre- and post-conditions. The notion of dominance studied in the current paper focuses on the initial and final snapshots of the artifact as it passes through a workflow; in particular, workflow $\mathcal{W}_1$ dominates workflow $\mathcal{W}_2$ if for each execution of $\mathcal{W}_2$ on a given initial snapshot, there is an execution of $\mathcal{W}_1$ on that initial snapshot that yields the same final snapshot. This notion of dominance is in fact a parametric notion that depends on the characterization of the policies of the performers that execute the services invoked by the workflows. In this paper, we study in detail the case of "absolute dominance", in which this policy places no

constraints on the possible executions. Alternative policies, which are the topic of future studies, might require that the service executions be deterministic, or that they be generic in the sense of database queries.

This paper develops results concerning absolute dominance for several variations of the underlying workflow model, based primarily on logical structure of the underlying domain of values. In the case of bounded-length executions, deciding absolute dominance can be reduced to first-order logic reasoning, which yields decidability if the underlying logical structure is, the integers with addition ($+$) and order ($<$), the rationals with addition and order, or the real closed field. For the unbounded case, we show that dominance is decidable if the logical structure has no function symbols and permits quantifier elimination, but it is undecidable for the cases of integers, rationals, or reals mentioned above.

Additional decidability results are obtained by focusing on FO logic/structures that have equality, order, and no function symbols. We borrow techniques from the powerful framework of Datalog with order constraints [9] to obtain decidability of absolute dominance in this case. In particular, we show decidability of absolute dominance for the cases of the integers with discrete order, and for rationals and reals with dense order. In all of these cases, we show that decidability is in exponential time.

Organizationally, Section 2 defines the formal model of artifact-centric workflow used in the paper, Section 3 defines the general notion of dominance, and absolute dominance in particular. Section 4 presents the theoretical results and Section 5 provides brief conclusions.

2 Artifact-Centric Workflows

In this paper we make use of a specific form of artifact-centric workflows, which we introduce in the following. We use a first-order logic $\mathcal{L}$ with equality, with predicates, constants, and possibly function symbols, over interpreted structures with a non-empty, possibly infinite domain. Examples of such interpreted structures are a dense or discrete order ($<$), Presburger arithmetic, or a real (closed) field. We will denote a structure $\mathbb{S}$ with domain $\Delta^{\mathbb{S}}$ over function symbols $f_1, f_2, \ldots$, and predicate symbols $p_1, p_2 \ldots$, as $\mathbb{S} = (\Delta^{\mathbb{S}}, f_1, f_2 \ldots, p_1, p_2 \ldots)$. We will assume to have one constant for each element of $\Delta^{\mathbb{S}}$, hence we usually omit constants (i.e., 0-ary functions) in the list of function symbols.

We assume an infinite alphabet $\mathcal{AN}$ of *attribute names*, which are also used as variables in logic formulas in $\mathcal{L}$ interpreted over $\mathbb{S}$. In the following, we use a, b for attribute names, c, d for values from the domain, and x, y, z for generic variables in formulas, all possibly with subscripts. Given a structure $\mathbb{S}$ with domain $\Delta^{\mathbb{S}}$, an *attribute-map* (w.r.t. $\mathbb{S}$), shortened as *amap*, is a total function from a finite set $X \subseteq \mathcal{AN}$ of attribute names into $\Delta^{\mathbb{S}} \cup \{\perp\}$, where the special symbol $\perp$ (which is not in $\Delta^{\mathbb{S}}$) has the intended meaning that the attribute is "undefined". Specifically, to represent that an attribute has an undefined value in an amap, i.e., has value equal to $\perp$, in formulas we shall use a syntactic shorthand: we introduce for each attribute name a an additional attribute $\bar{a}$,

and use $\bar{a} = 0$ to indicate that a is undefined, where 0 (zero) is a distinguished constant of $\mathcal{L}$. Then, in formulas, we write $a = \perp$ as a shorthand for $\bar{a} = 0$ and, e.g., $R(a_1, \ldots, a_n)$ as a shorthand for $\bar{a}_1 \neq 0 \wedge \cdots \wedge \bar{a}_n \neq 0 \wedge R(a_1, \ldots, a_n)$, where R is any predicate symbol, including equality.

Definition 1. *An* artifact (data schema) *is a non-empty set $\mathcal{A}$ of attribute names. The names are partitioned into the following three sets:*
- *the set $I_\mathcal{A}$ of* input *attributes,*
- *the set $T_\mathcal{A}$ of* temporary *attributes, and*
- *the nonempty set $O_\mathcal{A}$ of* output *attributes.*

A snapshot *σ of $\mathcal{A}$ is an amap whose domain is $\mathcal{A}$. σ is* initial *if $\sigma(a) \neq \perp$ for each $a \in I_\mathcal{A}$ and $\sigma(a) = \perp$ for each $a \in \mathcal{A} \setminus I_\mathcal{A}$. σ is* complete *if $\sigma(a) \neq \perp$ for each $a \in O_\mathcal{A}$.*

If $\mathcal{A}$ is an artifact, σ a snapshot of $\mathcal{A}$, and $X \subseteq \mathcal{A}$, then the *projection* of σ onto X, denoted $\sigma|_X$, is the function from X to $\Delta^\mathbb{S}$ defined by $\sigma|_X(x) = \sigma(x)$, for each $x \in X$.

A fundamental concept in the workflow model is that an artifact gets modified by a service in a single step (see later). To model this, we also allow *primed* attribute names as variables in our logic, which are used to represent the artifact after the modification. Let φ be a formula in which each free variable is an unprimed or primed attribute name. Let X contain the set of attribute names that occur unprimed in φ and Y' contain the set of attribute names that occur primed in φ. Let σ_X be an amap with domain X and $\sigma_{Y'}$ be an amap with domain Y'. In this case, $(\sigma_X, \sigma_{Y'})$ *models* φ *in* $\mathbb{S}$, denoted $(\sigma_X, \sigma_{Y'}) \models_\mathbb{S} \varphi$ if φ is true in $\mathbb{S}$ under the assignment that maps each unprimed attribute name $x \in X$ to $\sigma_X(x)$, and each primed attribute name $y' \in Y'$ to $\sigma_{Y'}(y')$.

It is clear that the notion of a pair (σ, σ') of snapshots over an artifact $\mathcal{A}$ modeling a formula φ can be used whenever the set of attribute names occurring in φ is contained in $\mathcal{A}$. If one thinks of σ as a snapshot preceding σ', then we are following the tradition of using unprimed variables to indicate a "current" state and primed variables to indicate a "next" state.

We introduce now services, which are the atomic units that progress a system.

Definition 2. *Given an artifact $\mathcal{A}$, a* service *(*specification*) for $\mathcal{A}$ is a 4-tuple $S = (I_S, O_S, \delta_S, \xi_S)$ where*

- *$I_S \subseteq I_\mathcal{A} \cup T_\mathcal{A}$ is called the* input *of S.*
- *$O_S \subseteq T_\mathcal{A} \cup O_\mathcal{A}$ is called the* output *of S.*
- *δ_S, the* pre-condition *of S, is a formula in $\mathcal{L}$ where each free variable is an unprimed attribute name from I_S.*
- *ξ_S, the* post-condition *of S, is a formula in $\mathcal{L}$ where each free variable is an unprimed attribute name from I_S or a primed attribute name from O_S.*

The frame formula *of S is the formula $\Phi_S \equiv \bigwedge_{a \in \mathcal{A} \setminus O_S} (a = a')$.*

Intuitively, we require that a service takes its inputs either from the inputs to the workflow (i.e., the input attributes $I_\mathcal{A}$ of the artifact), or from the temporary

attributes $T_{\mathcal{A}}$ of the artifact. The latter can be written by a service, together with the outputs of the workflow (i.e., the output attributes $O_{\mathcal{A}}$ of the artifact).

Definition 3. *Given an artifact $\mathcal{A}$ and a service $S = (I_S, O_S, \delta_S, \xi_S)$ for $\mathcal{A}$, an* execution *of S is a pair (σ, σ') of snapshots of $\mathcal{A}$ such that $(\sigma, \sigma') \models_{\mathbb{S}} \delta_S \wedge \xi_S \wedge \Phi_S$.*

Note that σ is in fact independent from σ' and that the frame formula Φ_S requires that in an execution (σ, σ') of S, each attribute not in O_S has in σ' the same value that it had in σ. Also, executions are in general *non-deterministic*, i.e., a service may have two executions (σ, σ') and (σ, σ''), with $\sigma' \neq \sigma''$.

We are interested in sequences of service executions that, from an initial snapshot of an artifact $\mathcal{A}$ may lead to a complete snapshot, i.e., one where all output attributes of $\mathcal{A}$ are defined. To formalize this notion, we first introduce the notion of *(artifact-centric) pre-workflow (schema)*, which is simply a pair $\mathcal{P} = (\mathcal{A}, \mathcal{S})$, where $\mathcal{S}$ a finite set of services for the artifact $\mathcal{A}$. Each (initial, complete) snapshot of $\mathcal{A}$ is also an (initial, complete, resp.) snapshot of $\mathcal{P}$. We can then provide the following definition of enactment.

Definition 4. *Given a pre-workflow $\mathcal{P} = (\mathcal{A}, \mathcal{S})$, an* enactment *$E$ of $\mathcal{P}$ (of* length *n) is a sequence*

$$\sigma_0, S_1, \sigma_1, \ldots, S_n, \sigma_n$$

where

- *σ_0 is an initial snapshot of $\mathcal{P}$,*
- *σ_i is a snapshot of $\mathcal{P}$, for $i \in [1..n]$,*
- *$S_i \in \mathcal{S}$, for $i \in [1..n]$,*
- *(σ_{i-1}, σ_i) is an execution of S_i, for $i \in [1..n]$.*

The enactment E is complete *if σ_n is complete. The* I/O-pair *of a complete enactment E is the pair $IO(E) = (\sigma_0|_{I_{\mathcal{A}}}, \sigma_n|_{O_A})$.*

Observe that $\sigma_0|_{I_{\mathcal{A}}} = \sigma_n|_{I_A}$ since the input attributes $I_{\mathcal{A}}$ of the artifact cannot be changed by services (see Definition 2)

We now introduce business rules, which specify the conditions under which a service may be executed. Given a pre-workflow $\mathcal{P} = (\mathcal{A}, \mathcal{S})$, a *(business) rule* ρ for $\mathcal{P}$ is an expression of the form "(**if** α **allow** S)" where α is a formula in which each free variable is an unprimed attribute name from $\mathcal{A}$, and $S \in \mathcal{S}$. With this notion in place, we are ready to provide the definition of artifact-centric workflows.

Definition 5. *Given a structure $\mathbb{S}$, a(n)* (artifact-centric) workflow (schema) *over $\mathbb{S}$ is a triple $\mathcal{W} = (\mathcal{A}, \mathcal{S}, \mathcal{R})$ where $(\mathcal{A}, \mathcal{S})$ is a pre-workflow and $\mathcal{R}$ is a finite set of rules for $(\mathcal{A}, \mathcal{S})$. An* enactment *of $\mathcal{W}$ is an enactment of its pre-workflow $(\mathcal{A}, \mathcal{S})$*

$$\sigma_0, S_1, \sigma_1, \ldots, S_n, \sigma_n$$

*such that for each $i \in [1..n]$, there is a rule "(**if** α **allow** S_i)" in $\mathcal{R}$ where $\sigma_{i-1} \models_{\mathbb{S}} \alpha$.*

In the following, we will omit the specification of the structure $\mathbb{S}$ when it is clear from the context.

3 Dominance

Intuitively, services are executed by a *performer*. Performers are often humans, but they could also be software components. In this paper we do not address different "roles" that different performers might have. Performers choose how a service is executed. Indeed, as mentioned for a given service S and a snapshot σ there may be executions (σ, σ') and (σ, σ''), with $\sigma' \neq \sigma''$. This non-determinism corresponds to the possibility that the performer may execute the same service on the same inputs differently. Intuitively, this might be because the performer is exercising human judgement, or because there is information about the snapshot σ that is not modeled within the formal system, or both.

In order to capture formally the above intuitions on performers, we introduce the notion of a "performance policy", which specifies the possible behaviors of the performers of the services of a workflow. Given a set of attributes X, We denote with $\mathcal{M}[X]$ the set of amaps over X.

Definition 6. *A* performance policy *for a workflow $\mathcal{W} = (\mathcal{A}, \mathcal{S}, \mathcal{R})$ is a function π whose domain is $\mathcal{S}$. The value of π on $S{=}(I_S, O_S, \delta_S, \xi_S) \in \mathcal{S}$, denoted $\pi[S]$, is a subset of $\mathcal{M}[I_S] \times \mathcal{M}[O_S]$ such that, if $(\mu, \nu) \in \pi[S]$, then $\mu \models_{\mathbb{S}} \delta_S$ and $(\mu, \nu) \models_{\mathbb{S}} \xi_S$. An execution (σ, σ') of S is* compliant *with π if $(\sigma|_{I_S}, \sigma'|_{O_S}) \in \pi[S]$. An enactment of $\mathcal{W}$ is* compliant *with π if each execution in the enactment is compliant with π.*

With the notion of performance policy at hand, we can now compare two artifact-centric workflows. In particular, we are interested in comparing two workflows in terms of how values for the input attributes are mapped into values for the output attributes. Also, we ignore the order in which the output attributes are written in one enactment versus the other enactment. For this, we say that two workflows $\mathcal{W}_1 = (\mathcal{A}_1, \mathcal{S}_1, \mathcal{R}_1)$ and $\mathcal{W}_2 = (\mathcal{A}_2, \mathcal{S}_2, \mathcal{R}_2)$ are *compatible* if $I_{\mathcal{A}_1} = I_{\mathcal{A}_2}$ and $O_{\mathcal{A}_1} = O_{\mathcal{A}_2}$. Note that the temporary attributes ($T_{\mathcal{A}_1}$ and $T_{\mathcal{A}_2}$) may be different.

Definition 7. *Let $\mathcal{W}_1 = (\mathcal{A}_1, \mathcal{S}_1, \mathcal{R}_1)$ and $\mathcal{W}_2 = (\mathcal{A}_2, \mathcal{S}_2, \mathcal{R}_2)$ be two compatible workflows and Π a class of performance policies. Then $\mathcal{W}_1$ is Π-*dominated *by $\mathcal{W}_2$, denoted $\mathcal{W}_1 \preceq_\Pi \mathcal{W}_2$, if the following holds. For each perfomance policy $\pi_1 \in \Pi$ for $\mathcal{W}_1$ there exists a performance policy $\pi_2 \in \Pi$ for $\mathcal{W}_2$ such that: for every enactment E_1 of $\mathcal{W}_1$ compliant with π_1 there is an enactment E_2 of $\mathcal{W}_2$ compliant with π_2 such that $IO(E_1) = IO(E_2)$.*

We consider also the case where we compare two workflows only w.r.t. enactments of bounded length. To this purpose we introduce the notion of *k-dominance* between two compatible workflows, denoted $\mathcal{W}_1 \preceq^k_\Pi \mathcal{W}_2$, whose definition is analogous to the one above, except that we consider only enactments whose length is $\leqslant k$.

This framework permits us to study a variety of behaviours of performers, i.e., of performance policies, including, e.g., policies where $\pi[S]$ is required to satisfy certain properties, such as being computable or tractable. In this paper

we concentrate on the most general performance policy, which states that the performers may use *any* execution of a service within the workflow (i.e., any execution that satisfies the pre- and post-conditions of the service), without further restrictions. We call this notion of dominance *absolute dominance*.

4 Absolute Dominance

We study now the problem of checking absolute dominance and absolute k-dominance. Let Abs denote the class of all performance policies. We say that $\mathcal{W}_1$ is *(k-)dominated absolutely* by $\mathcal{W}_2$ if $\mathcal{W}_1 \preceq_{Abs} \mathcal{W}_2$ (resp., $\mathcal{W}_1 \preceq^k_{Abs} \mathcal{W}_2$).

4.1 Enactments of Bounded Length

We deal first with the case of bounded absolute dominance, and show that we can characterize in a closed form the set of realizable I/O-pairs.

Lemma 1. *Let $\mathcal{W} = (\mathcal{A}, \mathcal{S}, \mathcal{R})$ be a workflow with service pre- and post-conditions and rule conditions expressed in FOL with equality. Let k be a positive integer. Then there is a FOL formula $\Psi^k_{\mathcal{W}}$ whose free variables are the input ($I_{\mathcal{A}}$) and output ($O_{\mathcal{A}}$) attributes of $\mathcal{A}$, that characterizes the set of all I/O-pairs of complete enactments of $\mathcal{W}$ compliant with Abs, for enactments whose length is bounded by k.*

Proof. We consider all possible sequences of services (possibly with repetitions) that may appear in enactments of length $n \leqslant k$, and characterize their I/O-pairs by means of a FOL formula.

Let $p = S_1, \ldots, S_n$ be such a sequence of services. Then, for $i \in [1..n]$, let

$$\begin{aligned}
\alpha^p_i &= \Big(\bigvee_{(\textbf{if } \alpha \textbf{ allow } S_i) \in \mathcal{R}} \alpha\Big)\, [a/a^{i-1} \mid a \in \mathcal{A}],\\
\delta^p_i &= \delta_{S_i}[a/a^{i-1} \mid a \in \mathcal{A}],\\
\xi^p_i &= \xi_{S_i}[a/a^{i-1} \mid a \in I_{S_i}][a'/a^i \mid a \in O_{S_i}],\\
\Phi^p_i &= \Phi_{S_i}[a/a^{i-1} \mid a \in (\mathcal{A} \setminus O_{S_i})][a'/a^i \mid a \in (\mathcal{A} \setminus O_{S_i})],
\end{aligned}$$

where $\varphi[a/b \mid a \in X]$ denotes the formula obtained from φ by renaming each (occurrence of) attribute $a \in X$ to b. Using such formulas, we build inductively, for each $i \in [0..n]$, the "cumulative post-condition", denoted $\hat{\xi}^p_i$ as follows:

- $\hat{\xi}^p_i = \mathit{true}$,
- $\hat{\xi}^p_i = \exists\{a^{i-1} \mid a \in \mathcal{A}\}(\hat{\xi}^p_{i-1} \wedge \alpha^p_i \wedge \delta^p_i \wedge \xi^p_i \wedge \Phi^p_i)$, for $i \in [1..n]$.

Note that $\hat{\xi}^p_i$, for $i \in [1..n]$, is a formula whose free variables are among $\{a^i \mid a \in \mathcal{A}\}$. It remains to project away the temporary attributes of the last step, and to impose that all output attributes are defined. Hence, we define

$$\Psi^p_{\mathcal{W}} = \Big(\exists\{a^n \mid a \in T_{\mathcal{A}}\}(\hat{\xi}^p_n \wedge \textstyle\bigwedge_{a \in O_{\mathcal{A}}} a^n \neq \bot)\Big)\, [a^n/a \mid a \in I_{\mathcal{A}} \cup O_{\mathcal{A}}].$$

By quantifying over all possible sequences of services of length up to k, we obtain the desired formula

$$\Psi^k_{\mathcal{W}} = \bigvee_{\substack{S_1,\ldots,S_n,\\ \text{for } S_i \in \mathcal{S}, i \in [1..n], n \leqslant k}} \Psi^{S_1,\ldots,S_n}_{\mathcal{W}}.$$

It is not difficult to prove by induction on k that $\Psi^k_{\mathcal{W}}$ characterizes the set of all I/O-pairs of complete enactments of $\mathcal{W}$ compliant with Abs, for enactments whose length is bounded by k. □

Using the above characterization of I/O-pairs, we can determine absolute k-dominance $\mathcal{W}_1 \preceq^k_{Abs} \mathcal{W}_2$ between two compatible workflows $\mathcal{W}_1 = (\mathcal{A}_1, \mathcal{S}_1, \mathcal{R}_1)$ and $\mathcal{W}_2 = (\mathcal{A}_2, \mathcal{S}_2, \mathcal{R}_2)$, where $I = I_{\mathcal{A}_1} = I_{\mathcal{A}_2}$ and $O = O_{\mathcal{A}_1} = O_{\mathcal{A}_2}$, by simply checking whether the following formula is true in $\mathbb{S}$:

$$\forall\{a \in I \cup O\}(\Psi^k_{\mathcal{W}_1} \rightarrow \Psi^k_{\mathcal{W}_2}). \qquad (\star)$$

Hence, in all those cases where FOL over $\mathbb{S}$ is decidable, we obtain decidability of absolute k-dominance for workflows over $\mathbb{S}$.

Theorem 1. *For each positive integer k, absolute k-dominance between workflows over $\mathbb{S}$ is decidable for the following structures:*

1. $(\mathbb{Z}, +, <)$*, integers with additions.*
2. $(\mathbb{Q}, +, <)$*, rational numbers with additions.*
3. $(\mathbb{R}, +, \times, <)$*, real numbers with additions and multiplications (the real closed field).*

Proof (Sketch). By Lemma 1, absolute k-dominance between two workflows holds if and only if the formula shown in Equation $(\star)$ is true in the underlying structure. Thus the decidability results follow immediately from the decidability results for Presburger arithmetic [13] (Case 1) and the real closed field [17] (Cases 2 and 3). □

We discuss briefly the complexity of the decisions problems. Given two workflows of length ℓ, the formula in Equation $(\star)$ has length at most $O(k^{k+1}\ell)$. For the domain of integers with additions, since the complexity of Presburger arithmetic is double exponential [6], it follows that the absolute k-dominance problem has complexity double expential in ℓ and triple exponential in k. On the other hand, since the complexity of the FO theory for the real closed field is exponential [1,14], the dominance problem is exponential in ℓ and double exponential in k. Note that the above analysis puts coarse upper bounds in the most general situations. If we focus on restricted classes, such as services only having quantifier free formulas as pre- and post-conditions, and put bounds on the number of temporary variables they can use, the complexity upper bounds can be refined.

$$
\begin{aligned}
\mathit{ioPairs}(I, O_1) &\leftarrow \mathit{initial}(I, T, O), \mathit{transStar}(I, T, O, I_1, T_1, O_1), \\
&\quad\ \mathit{complete}(I_1, T_1, O_1). \\
\mathit{initial}(I, T, O) &\leftarrow \mathit{defined}(I), \mathit{undefined}(T, O). \\
\mathit{complete}(I, T, O) &\leftarrow \mathit{defined}(O). \\
\mathit{transStar}(I, T, O, I, T, O). & \\
\mathit{transStar}(I, T, O, I_2, T_2, O_2) &\leftarrow \mathit{trans}(I, T, O, I_1, T_1, O_1), \\
&\quad\ \mathit{transStar}(I_1, T_1, O_1, I_2, T_2, O_2). \\
\mathit{trans}(I, T, O, I_1, T_1, O_1) &\leftarrow \mathit{transBy}S_1(I, T, O, I_1, T_1, O_1). \\
&\ \ \ldots \\
\mathit{trans}(I, T, O, I_1, T_1, O_1) &\leftarrow \mathit{transBy}S_n(I, T, O, I_1, T_1, O_1). \\
\mathit{transBy}S_i(I, T, O, I_1, T_1, O_1) &\leftarrow \delta_{S_i}(I, T, O), \mathit{rulesAllow}S_i(I, T, O), \\
&\quad\ \mathit{nextSnapshotBy}S_i(I, T, O, I_1, T_1, O_1). \\
\mathit{nextSnapshotBy}S_i(I, T, O, I_1, T_1, O_1) &\leftarrow \xi_{S_i}(I, T, O, I_1, T_1, O_1), \Phi_{S_i}(I, T, O, I_1, T_1, O_1). \\
\mathit{rulesAllow}S_i(I, T, O) &\leftarrow \alpha_i^1(I, T, O). \\
&\ \ \ldots \\
\mathit{rulesAllow}S_i(I, T, O) &\leftarrow \alpha_i^{m_i}(I, T, O).
\end{aligned}
$$

Fig. 1. Constraint Datalog program $P_{\mathcal{W}}$ capturing the I/O-pairs of a workflow $\mathcal{W}$ for enactments of unbounded length

4.2 Enactments of Unbounded Length

We now turn to enactments of unbounded length. Oone might think that when the FO logic over the structure $\mathbb{S}$ admits quantifier elimination, the characterization in Lemma 1 could be extended to enactments of unbounded length. In general, it is not clear how this can be possibly done. In fact, the following can be established.

Theorem 2. *Absolute dominance between two workflows is undecidable for the following structures:*

1. $(\mathbb{Z}, +, <)$*, integers with additions.*
2. $(\mathbb{Q}, +, <)$*, rational numbers with additions.*
3. $(\mathbb{R}, +, \times, <)$*, real numbers with additions and multiplications (the real closed field).*

Proof (Sketch). The proofs are accomplished by reductions from Hilbert's 10th problem (computing integer roots of polynomials with integer coefficients), which is known to be undecidable (see [10]). Roughly, the idea of the reduction is to guess potential (integer) solutions (with a simple increment service) and then verify if they are indeed solutions. Over the given structures, one can easily compute multiplications with repeated additions. Thus, the verfication can also be expressed with a workflow. □

In the following, we explore more restricted FO logic/structures that consists of equality and order and without any functions. We borrow techniques from the

powerful framework of Datalog with order constraints [9] to show that absolute dominance can still be decidable for these structures.

Specifically, we focus on workflows whose service pre- and post-conditions and rule conditions are quantifier free formulas over equality ($=$) and order ($<$) constraints. Given a workflow $\mathcal{W} = (\mathcal{A}, \mathcal{S}, \mathcal{R})$, we construct a constraint Datalog program $P_{\mathcal{W}}$ as shown in Figure 1, which views service pre- and post-conditions and rule conditions as constraint relations [9]. In the specification of the program, we have assumed that $\mathcal{S} = \{S_1, \ldots, S_n\}$, and that for $i \in [1..n]$, (**if** α_i^1 **allow** S_i), ..., (**if** $\alpha_i^{m_i}$ **allow** S_i) are all rules having S_i as consequent. Such a program provides a characterization of the set of all I/O-pairs of $\mathcal{W}$ under Abs for enactments of unbounded length. We briefly comment on the rules of $P_{\mathcal{W}}$.

- $ioPairs(I, O_1) \leftarrow initial(I, T, O), transStar(I, T, O, I_1, T_1, O_1), complete(I_1, T_1, O_1)$.
  Generates all I/O-pairs of complete compliant enactments. Here, I, T, and O stand respectively for the input, temporary, and output attributes of the artifact in the initial snapshot. Similarly, I_1, T_1, and O_1 stand for the same attributes in the complete snapshot at the end of the enactment.
- $initial(I, T, O) \leftarrow defined(I), undefined(T, O)$.
  States when a snapshot is initial, i.e., all input attributes are defined, and all temporary and output attributes are undefined.
- $complete(I, T, O) \leftarrow defined(O)$.
  States when a snapshot is complete, i.e., all output attributes are defined.
- $transStar(I, T, O, I, T, O)$.
  $transStar(I, T, O, I_2, T_2, O_2) \leftarrow trans(I, T, O, I_1, T_1, O_1), transStar(I_1, T_1, O_1, I_2, T_2, O_2)$.
  Compute the reflexive transitive closure of *trans*, defined below.
- $trans(I, T, O, I_1, T_1, O_1) \leftarrow transByS_1(I, T, O, I_1, T_1, O_1)$.
  $\ldots$
  $trans(I, T, O, I_1, T_1, O_1) \leftarrow transByS_n(I, T, O, I_1, T_1, O_1)$.
  State that transition can be made by (and only by) services.
- $transByS_i(I, T, O, I_1, T_1, O_1) \leftarrow \delta_{S_i}(I, T, O), rulesAllowS_i(I, T, O), nextSnapshotByS_i(I, T, O, I_1, T_1, O_1)$.
  States that a transition is made by service S_i when its preconditions hold and the rules allow S_i to execute. The service produces the next snapshot.
- $nextSnapshotByS_i(I, T, O, I_1, T_1, O_1) \leftarrow \xi_{S_i}(I, T, O, I_1, T_1, O_1), \Phi_{S_i}(I, T, O, I_1, T_1, O_1)$.
  States that the execution of S_i produces the next snapshot on the basis of the service post-conditions and its frame formula. Note that ξ_{S_i} and Φ_{S_i} together constrain all variables I_1, T_1, O_1 w.r.t. I, T, and O, either with effect ξ_{S_i} or with the frame formula Φ_{S_i}.
- $rulesAllowS_i(I, T, O) \leftarrow \alpha_i^1(I, T, O)$.
  $\ldots$
  $rulesAllowS_i(I, T, O) \leftarrow \alpha_i^{m_i}(I, T, O)$.
  State that S_i is allowed to be executed.

We can show the following property regarding $P_{\mathcal{W}}$.

Lemma 2. *Given a workflow $\mathcal{W} = (\mathcal{A}, \mathcal{S}, \mathcal{R})$ over $\mathbb{S}$, let $\mathcal{P}_{\mathcal{W}}$ be the constraint Datalog program constructed from $\mathcal{W}$ as specified above. Then an I/O-pair (σ_I, σ_O) is an I/O-pair of a complete enactment of $\mathcal{W}$ if and only if (σ_I, σ_O) is returned by the above constraint Datalog program when it is evaluated over $\mathbb{S}$.*

Proof (Sketch). The proof can be done via an induction argument on the length of the enactment producing the I/O-pair. □

We note here that the above construction is rather general, and that the resulting Datalog program may not always terminate when the logic language includes at least one function symbol, regardless of whether the language/structure admits quantifier elimination.

We now exploit results on constraint Datalog with order constraints that state, for some specific structures $\mathbb{S}$, that a constraint Datalog program P can be evaluated in closed form over $\mathbb{S}$ to produce a FOL formula φ_P over $\mathbb{S}$ (with the output variables of P as free variables). The resulting FOL formula is in fact equivalent to the Datalog program [9,15]. Specifically, from the results in [9,15] it follows that the program $P_{\mathcal{W}}$ is equivalent to a formula of $\mathcal{L}$ over the structure $\mathbb{S}$ having I and O_1 as free variables, in the cases where the logic $\mathcal{L}$ is FOL with equality, and $\mathbb{S}$ is a dense order over the rationals or reals (with all rationals as constants), or a linear order over the integers (with all integers as constants). Hence, extending Lemma 2, we obtain the following result.

Lemma 3. *Let $\mathcal{W} = (\mathcal{A}, \mathcal{S}, \mathcal{R})$ be a workflow over a structure $\mathbb{S}$ with service pre- and post-conditions and rule conditions expressed as quantifier-free formulas in FOL. For each of the following structures, there is a quantifier-free FOL-formula $\Psi_{\mathcal{W}}$ whose free variables are the input ($I_{\mathcal{A}}$) and output ($O_{\mathcal{A}}$) attributes of $\mathcal{A}$, that characterizes the set of all I/O-pairs of complete enactments of $\mathcal{W}$ compliant with Abs:*

- $(\mathbb{Z}, <)$, *integers with the discrete order.*
- $(\mathbb{Q}, <)$, *rational numbers with the dense order.*
- $(\mathbb{R}, <)$, *real numbers with the dense order.*

Proof (Sketch). Clearly, for each service S, we can construct a constraint relation (quantifier-free formula in disjunctive normal form) [9] that represents its set of input and output pairs allowed by the pre- and post-conditions. Similarly, each rule condition can also be represented as a constraint relation. Let the constraint database consist of the constraint relations representing services and rule conditions. The Datalog program constructed above can then be evaluated as a query against the constraint database. Results from [9,15] state that the query answer can be computed effectively and represented as a constraint relation. The constraint relation is in fact a quantifier-free FOL formula. □

For each structure listed in Lemma 3, we can proceed as for the case of bounded enactments, and exploit the formulas $\Psi_{\mathcal{W}_1}$ and $\Psi_{\mathcal{W}_2}$ to rephrase, also for unbounded enactments, absolute dominance between two workflows $\mathcal{W}_1$ and $\mathcal{W}_2$ over $\mathbb{S}$ in terms of evaluation over $\mathbb{S}$ of the FOL formula

$$\forall\{a \in I \cup O\}(\Psi_{\mathcal{W}_1} \rightarrow \Psi_{\mathcal{W}_2}).$$

From the decidability of FOL with equality over $\mathbb{S}$, we get the following result.

Theorem 3. *Absolute dominance between workflows over* $\mathbb{S}$ *is decidable in the following cases:*

- $(\mathbb{Z}, <)$*, integers with discrete order.*
- $(\mathbb{Q}, <)$*, rational numbers with dense order.*
- $(\mathbb{R}, <)$*, real numbers with dense order.*

The argument for the above theorem is similar to Theorem 1. We now briefly discuss the complexity of the above decision problems. Note that the query evaluation of the Datalog program can be done in exponential time and the size of the constraint relations that represent all possible enactments are of exponential size in the terms of the input workflow [9,15]. Applying known complexity of results in logic, checking the FOL formula that characterizes the dominance would add one additional level of exponentiation (in the cases of $(\mathbb{R}, <)$ and $(\mathbb{Q}, <)$) or two additional levels of exponentiation (in the case of $(\mathbb{Z}, <)$). However, this complexity for all three cases can be improved to overall single exponential time; we outline the algorithms in the following.

We call a conjunction of constraints *primitive* if it is satisfiable but not logically implied by but not equivalent to another conjunction. It is easy to see that for a given (finite) number of variables and a given finite set of constants, the number of pairwise non-equivalent primitive constraints is also finite (but exponential in terms of the total number of variables and constants.

For a given pair of workflows $\mathcal{W}_1, \mathcal{W}_2$, let V be the set of constants occurring in either $\mathcal{W}_1$ or $\mathcal{W}_2$. We convert the results of the Datalog program for $\mathcal{W}_1$ and $\mathcal{W}_2$ into two sets of primitive conjunctions of constraints, for $\mathcal{W}_1$ and $\mathcal{W}_2$, resp., of form "$x\theta v$" or "$x\theta y$" where v is in V and θ is either "=" or "<". We then remove each unsatisfiable primitive conjunction, which can be done in PTIME [8]. It can be shown that dominance holds for the workflows iff the containment of two sets of primitive conjunctions holds. Since the number of primitive conjunctions is exponential in the size of input, so is the complexity of the algorithms.

5 Conclusions

In this paper we have addressed the problem of comparing artifact-centric workflows by introducing a general notion of dominance between workflows. Such a notion is parametric with respect to a class of policies adopted by the performers of the services that are invoked by the workflow. Here we have focused on the most general type of performers, which may use any execution of a service

within the workflow, resulting in the notion of "absolute dominance". We and have provided decidability and complexity results for this case.

The framework and results reported in this paper provide a basis and starting point for a rich study of dominance between artifact-centric workflows, and leave many questions yet to be explored. As noted above, our notion of dominance is focused only on the initial and final snapshots of a workflow execution; it would be useful to understand a richer notion of dominance that incorporates the order in which output values of the workflow execution are created. Also, the model used here assumes that all relevant data is held within the artifact. A useful extension would be to study the natural case in which there is also an external, basically fixed database that the conditions can refer to (for example, in the spirit of [5]). It is also of interest to study other types of performers, characterized by restrictions on the policy they may adopt. A notable case is the one where the performance policy is a deterministic function from the input attributes to the output attributes of the service. In other words, a performer deterministically takes its decision considering only the values of the input attributes of the service it is executing, and hence, if it re-executes a services with the same inputs, it takes the same decision, producing the same outputs, as in the previous execution.

References

1. Ben-Or, M., Kozen, D., Reif, J.: The complexity of elementary algebra and geometry. J. of Computer and System Sciences 32(2), 251–264 (1986)
2. Bhattacharya, K., Caswell, N.S., Kumaran, S., Nigam, A., Wu, F.Y.: Artifact-centered operational modeling: Lessons from customer engagements. IBM Systems Journal 46(4), 703–721 (2007)
3. Bhattacharya, K., et al.: A model-driven approach to industrializing discovery processes in pharmaceutical research. IBM Systems Journal 44(1), 145–162 (2005)
4. Bhattacharya, K., Gerede, C., Hull, R., Liu, R., Su, J.: Towards formal analysis of artifact-centric business process models. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 288–304. Springer, Heidelberg (2007)
5. Deutsch, A., Hull, R., Patrizi, F., Vianu, V.: Automatic verification of data-centric business processes. In: Proc. of the 12th Int. Conf. on Database Theory (ICDT 2009), pp. 252–267 (2009)
6. Fischer, M.J., Rabin, M.O.: Super-exponential complexity of Presburger arithmetic. In: SIAM-AMS Proceedings, vol. 7, pp. 27–41 (1974)
7. Fritz, C., Hull, R., Su, J.: Automatic construction of simple artifact-based workflows. In: Proc. of the 12th Int. Conf. on Database Theory (ICDT 2009), pp. 225–238 (2009)
8. Guo, S., Sun, W., Weiss, M.A.: Solving satisfiability and implication problems in database systems. ACM Trans. on Database Systems 21(2), 270–293 (1996)
9. Kanellakis, P.C., Kuper, G.M., Revesz, P.Z.: Contraint query languages. J. of Computer and System Sciences 51, 26–52 (1995)
10. Matiyasevich, Y.: Hilbert's 10th Problem. The MIT Press, Cambridge (1993)
11. Narayanan, S., McIlraith, S.: Simulation, verification and automated composition of web services. In: Proc. of the 11th Int. World Wide Web Conf. WWW 2002 (2002)

12. Nigam, A., Caswell, N.S.: Business artifacts: An approach to operational specification. IBM Systems Journal 42(3), 428–445 (2003)
13. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: Comptes rendus du premier Congrès des Mathématiciens des Pays Slaves, Warszawa, pp. 92–101 (1929)
14. Renegar, J.: On the computational complexity and geometry of the first-order theory of the reals. Journal of Symbolic Computation 13, 255–352 (1992)
15. Revesz, P.Z.: A closed-form evaluation for Datalog queries with integer (gap)-order constraints. Theoretical Computer Science 116, 117–149 (1993)
16. Strosnider, J., Nandi, P., Kumarn, S., Ghosh, S., Arsanjani, A.: Model-driven synthesis of SOA solutions. IBM Systems Journal 47(3), 415–432 (2008)
17. Tarski, A.: A Decision Method for Elementary Algebra and Geometry. University of California Press, Berkeley (1951)