SMT-Based Safety Verification of Data-Aware Processes under Ontologies (Preliminary Results)

Diego Calvanese[1,2], Alessandro Gianola[1], Andrea Mazzullo[1], and Marco Montali[1]

[1] KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy
surname@inf.unibz.it
[2] Computing Science Department, Umeå University, Sweden

Abstract. In the context of verification of data-aware processes, a formal approach based on *satisfiability modulo theories* (SMT) has been considered to verify parameterised safety properties of so-called *artifact-centric systems*. This approach requires a combination of model-theoretic notions and algorithmic techniques based on *backward reachability*. We introduce here a variant of one of the most investigated models in this spectrum, namely *simple artifact systems* (*SASs*), where, instead of managing a database, we operate over a description logic (DL) ontology expressed in (a slight extension of) RDFS. This DL, enjoying suitable model-theoretic properties, allows us to define DL-based SASs to which backward reachability can still be applied, leading to decidability in PSPACE of the corresponding safety problems.

1 Introduction

Verifying and reasoning about dynamic systems that integrate processes and data is a long-standing challenge that attracted considerable attention, and that led to a flourishing series of results, within business process management [28,9,20] and data management [30,8,4,17,18]. Among the several conceptual models studied in this area, data-centric systems and in particular artifact-centric systems have been brought forward as a principled approach where relevant (business) objects are elicited, then defining how actions evolve them throughout their lifecycle [24]. Different formal models have been proposed to capture artifact systems and study their verification [8]. One of the most studied settings considers artifact systems as being composed of: a *read-only database* storing background information about artifacts that does not change during the evolution of the system; a *working memory*, used to store data that can be modified in the course of the evolution; and *transitions* (also called *actions* or *services*) that query the

read-only database and the working memory and use the retrieved answers to update the working memory. Verification of such systems is particularly challenging, not only because the working memory in general evolves through infinitely many different configurations, but also because the desired verification properties should hold regardless of the specific content of the read-only database, thus calling for a particular form of parameterised verification [15,18,10,11].

In this paper, we study for the first time semantic artifact systems where the read-only database is substituted by a Description Logic ontology, which stores background, incomplete information about the artifacts. In this setting, two possible notions of parameterisation may be studied: one where the evolution of the system is verified against all possible choices for the ABox, another where verification is against all possible models of a fixed ABox. In this work, we adopt the latter hypothesis, and thus verify whether the artifact system enjoys desired properties irrespectively of how the information explicitly provided by the ABox is completed through the TBox assertions.

More in detail, we consider an extensively studied, basic model of such artifact-centric systems, called *simple artifact system* (*SAS*) in [11], where the artifact working memory consists of a fixed set of *artifact variables* [16,15,11]. On top of this basis, we study the verification of safety properties in the case where the ontology is specified in (a slight extension of) RDFS [6], a schema language for the Semantic Web formalized by the W3C, and we make use of the ontology signature to express the transitions that update the working memory. For this setting, we show that we can decide safety properties in PSpace by relying on an SMT-based backward reachability procedure.

In spirit, our approach is reminiscent of previous works studying the verification of dynamic systems (in particular, Golog programs) operating over a DL ontology, such as [14,31]. In fact, both in their settings and ours, the dynamic system evolves each model of the ontology, and verification properties are assessed over all the resulting evolutions. This is radically different from approaches where the ABox itself is evolved by the process, with an execution semantics following Levesque's functional approach, in which query entailment over the current state is used to compute the successor states [3,5]. However, we differ from [14,31] in that our goal is not only to derive foundational results, but also to transfer such results into practical algorithms and thus obtain a model that is readily implementable by relying on a state-of-the-art model checker such as MCMT [22]. As customary in the formal literature on artifact-centric systems, our approach is based on actions that manipulate the artifact variables, coupled with condition-action rules that declaratively define which actions are currently executable, and with which parameters. Alternative choices could be seamlessly taken, by adapting approaches that rely on an explicit description of the control-flow, e.g., based on state machines [26] or Petri nets interpreted with interleaving semantics [20,27].

Full details are provided in the extended version of this paper [12].

2 Preliminaries

In this section, we first recall the syntax and semantics of first-order logic (FO). We then define the syntax of the DL $RDFS_{+}$ considered in this paper, which is a slight extension of RDFS [6]. Its semantics is provided by means of the standard translation, mapping $RDFS_{+}$ ontologies into equivalent sets of FO formulas.

2.1 First-Order Logic Preliminaries

The alphabet of *first-order logic* (*FO*) consists of: countably infinite and pairwise disjoint sets $\mathsf{N_P}$ of *predicate symbols* (with $\mathsf{ar}(P) \in \mathbb{N}$ being the arity of $P \in \mathsf{N_P}$), $\mathsf{N_F}$ of *function symbols* (with $\mathsf{ar}(f) \in \mathbb{N}$ being the arity of $f \in \mathsf{N_F}$), $\mathsf{N_I}$ of *individual symbols* (or *individual names*), and Var of *variables*; the *equality symbol* '='; the *Boolean operators* '$\neg$' and '$\wedge$'; and the *existential quantifier* '$\exists$'. An (*FO*) *formula* is an expression $\varphi ::= P(\underline{t}) \mid s = t \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \exists x\varphi$, where $x \in \mathsf{Var}$, $P \in \mathsf{N_P}$, s, t are terms, and $\underline{t} = (t_1, \ldots, t_{\mathsf{ar}(P)})$ is a (possibly empty) tuple of terms, where *terms* are defined inductively as follows: $t ::= x \mid a \mid f(\underline{t})$, where $x \in \mathsf{Var}$, $a \in \mathsf{N_I}$, $f \in \mathsf{N_F}$, and $\underline{t} = (t_1, \ldots, t_{\mathsf{ar}(f)})$. A formula of the form $P(\underline{t})$ is called an *atom*, and a *literal* has the form $P(\underline{t})$ or $\neg P(\underline{t})$. We adopt the usual abbreviations and conventions: in particular, $\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi)$ and $\forall x\varphi = \neg\exists x\neg\varphi$, where $\forall$ is the *universal quantifier*. We write $\varphi(\underline{x})$ to indicate that the *free variables* (defined as usual) of φ are included in $\underline{x}$, and we write $\varphi(\underline{a})$ for the formula obtained from $\varphi(\underline{x})$ by substituting $\underline{a}$ to $\underline{x}$. Similar notions and notation are adopted for terms. A *sentence* is defined as a formula without free variables, while we call *quantifier-free* a formula without any occurrence of existential or universal quantifiers. A formula is *existential* if it has the form $\exists\underline{x}\varphi(\underline{x})$, where φ is a quantifier-free formula, and it is *universal* if it has the form $\forall\underline{x}\varphi(\underline{x})$, where φ is quantifier-free. A (*FO*) *theory* T is a set of FO sentences, and T is said to be *universal* if every $\varphi \in T$ is universal. A *signature* Σ is a subset of $\mathsf{N_P} \cup \mathsf{N_F} \cup \mathsf{N_I}$. For a set Γ of formulas, the *signature of* Γ, denoted Σ_Γ, is the set of predicate, function, and individual symbols occurring in Γ. Given a signature Σ, we say that Γ is a set of *Σ-formulas* if $\Sigma_\Gamma = \Sigma$ (we will use *Σ-formula*, *Σ-theory*, etc., in an analogous way).

An (*FO*) *interpretation* is a pair $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty set, called *domain* of $\mathcal{I}$, and $\cdot^\mathcal{I}$ is an *interpretation function* such that: $P^\mathcal{I} \subseteq (\Delta^\mathcal{I})^{\mathsf{ar}(P)}$, for every $P \in$ NP; $f^\mathcal{I} \colon (\Delta^\mathcal{I})^{\mathsf{ar}(f)} \longrightarrow \Delta^\mathcal{I}$, for every $f \in \mathsf{N_F}$; and $a^\mathcal{I} \in \Delta^\mathcal{I}$, for every $a \in \mathsf{N_I}$. An *assignment* in $\mathcal{I}$ is a function $\mathfrak{a} \colon \mathsf{Var} \longrightarrow \Delta^\mathcal{I}$. We define the *value* of a term t in $\mathcal{I}$ under $\mathfrak{a}$ as follows: $\mathfrak{a}(t) = \mathfrak{a}(x)$, if $t = x$; $\mathfrak{a}(t) = a^\mathcal{I}$, if $t = a \in \mathsf{N_I}$; and $\mathfrak{a}(t) = f^\mathcal{I}(\mathfrak{a}(\underline{t}))$, if $t = f(\underline{t})$, where $f \in \mathsf{N_F}$ and, for an m-tuple $\underline{t} = (t_1, \ldots, t_m)$ of terms, we set $\mathfrak{a}(\underline{t}) = (\mathfrak{a}(t_1), \ldots, \mathfrak{a}(t_m))$. The notion of a formula φ being *satisfied* in an interpretation $\mathcal{I}$ under an assignment

$\mathfrak{a}$, or of $\mathcal{I}$ being a *model* of φ under $\mathfrak{a}$, written $\mathcal{I} \models^{\mathfrak{a}} \varphi$, is inductively defined as:

$$\begin{array}{ll}
\mathcal{I} \models^{\mathfrak{a}} P(\underline{t}) & \text{iff } \mathfrak{a}(\underline{t}) \in P^{\mathcal{I}}, \\
\mathcal{I} \models^{\mathfrak{a}} s = t & \text{iff } \mathfrak{a}(s) = \mathfrak{a}(t), \\
\mathcal{I} \models^{\mathfrak{a}} \neg\psi & \text{iff not } \mathcal{I} \models^{\mathfrak{a}} \psi, \\
\mathcal{I} \models^{\mathfrak{a}} \psi \wedge \chi & \text{iff } \mathcal{I} \models^{\mathfrak{a}} \psi \text{ and } \mathcal{I} \models^{\mathfrak{a}} \chi, \\
\mathcal{I} \models^{\mathfrak{a}} \exists x\psi & \text{iff } \mathcal{I} \models^{\mathfrak{a}'} \psi \text{ for some } \mathfrak{a}' \text{ that can differ from } \mathfrak{a} \text{ on } x.
\end{array}$$

For a formula $\varphi(\underline{x})$, we write $\mathcal{I} \models \varphi[\underline{d}]$ in place of $\mathcal{I} \models^{\mathfrak{a}} \varphi(\underline{x})$, with $\mathfrak{a}(\underline{x}) = \underline{d}$. We say that a set Γ of formulas is *satisfied* in an interpretation $\mathcal{I}$ under an assignment $\mathfrak{a}$, or that $\mathcal{I}$ is a *model* of Γ under $\mathfrak{a}$, written $\mathcal{I} \models^{\mathfrak{a}} \Gamma$, if $\mathcal{I} \models^{\mathfrak{a}} \varphi$, for every $\varphi \in \Gamma$ (we refer to a singleton $\Gamma = \{\varphi\}$ simply as φ). For a sentence φ, the satisfaction of φ in $\mathcal{I}$ under $\mathfrak{a}$ does not depend on $\mathfrak{a}$, thus we write $\mathcal{I} \models \varphi$ in place of $\mathcal{I} \models^{\mathfrak{a}} \varphi$, and we say that φ is satisfied in $\mathcal{I}$. For a theory T, we say that T is *satisfied* in an interpretation $\mathcal{I}$ (or that $\mathcal{I}$ is a *model* of T), written $\mathcal{I} \models T$, if every sentence of T is satisfied in $\mathcal{I}$. A formula φ is *satisfiable w.r.t. T* (or *T-satisfiable*) if there exist an interpretation $\mathcal{I}$ and an assignment $\mathfrak{a}$ in $\mathcal{I}$ such that $\mathcal{I} \models T$ and $\mathcal{I} \models^{\mathfrak{a}} \varphi$. Moreover, we say that T *logically implies* a formula φ, or that φ is a *logical consequence* of T, written $T \models \varphi$, if, for every interpretation $\mathcal{I}$ and every assignment $\mathfrak{a}$ in $\mathcal{I}$, $\mathcal{I} \models T$ implies that $\mathcal{I} \models^{\mathfrak{a}} \varphi$. Finally, formulas φ, ψ are *equivalent w.r.t. T* (or *T-equivalent*) if $T \models \varphi \leftrightarrow \psi$.

2.2 Description Logics Preliminaries

Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be countably infinite and pairwise disjoint sets of *concept*, *role*, and *individual names*, respectively (with $\mathsf{N_C} \cup \mathsf{N_R} \subseteq \mathsf{N_P}$, i.e., concept and role names are predicate symbols, with arity 1 and 2, respectively).

The DL we consider here is an extension of RDFS [6] with disjointness between concepts and roles, conjunction and (one-level) qualified existential quantification on the left-hand side of inclusions, and inclusion of direct and inverse roles. We denote such DL *RDFS$_+$*, and we define it below.

In *RDFS$_+$*, *concepts* C and *roles* R are defined according to the grammar

$$\begin{array}{rcl}
R & ::= & P \mid P^-, \\
C & ::= & A_1 \sqcap \cdots \sqcap A_n \mid \exists R.\top \mid \exists R.A,
\end{array}$$

where $P \in \mathsf{N_R}$, $n \geq 1$, and $A, A_1, \ldots, A_n \in \mathsf{N_C}$.

A *concept inclusion (CI)* has the form $C \sqsubseteq A$ or $C \sqsubseteq \neg A$, and a *role inclusion (RI)* has the form $R \sqsubseteq R'$ or $R \sqsubseteq \neg R'$, where C is an *RDFS$_+$* concept, $A \in \mathsf{N_C}$, and R, R' are roles. An *RDFS$_+$ TBox* $\mathcal{T}$ is a finite set of CIs and RIs. An *assertion* has the form $A(a)$, $\neg A(a)$, $P(a, b)$, $\neg P(a, b)$, $(a = b)$, or $\neg(a = b)$, where $A \in \mathsf{N_C}$, $P \in \mathsf{N_R}$, and $a, b \in \mathsf{N_I}$. An *ABox* $\mathcal{A}$ is a finite set of assertions. (We point out that in an ABox we allow for negated assertions, which is a feature that is not always supported in DLs.) An *RDFS$_+$ ontology* $\mathcal{O}$ is a pair $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox.

We observe that *RDFS$_+$* is incomparable in expressive power with the DLs of the popular *DL-Lite* family [7,2]. Indeed, while *DL-Lite* allows for the use of

existential quantification on the right-hand side of CIs, these are ruled out in $RDFS_{+}$. On the other hand, in $RDFS_{+}$ one can locally type the second component of a role through the use of qualified existential quantification on the left-hand side of CIs, while this is not possible in *DL-Lite*. As we will see later, differently from what happens for *DL-Lite*, the FO translation of an $RDFS_{+}$ ontology is a universal theory.

Example 1. To represent part of the domain knowledge on job hiring processes for university personnel, we define the $RDFS_{+}$ ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ consists of the following concept inclusions:

$$\begin{aligned}
\mathsf{AcademicPosition} &\sqsubseteq \mathsf{JobPosition} & \mathsf{AcademicPosition} &\sqsubseteq \neg\mathsf{AdminPosition}\\
\mathsf{AdminPosition} &\sqsubseteq \mathsf{JobPosition} & \mathsf{User} &\sqsubseteq \neg\mathsf{JobPosition}\\
\exists\mathsf{appliesFor}.\top &\sqsubseteq \mathsf{User} & \exists\mathsf{appliesFor}^{-}.\top &\sqsubseteq \mathsf{JobPosition}\\
\exists\mathsf{suitableFor}.\top &\sqsubseteq \mathsf{User} & \exists\mathsf{suitableFor}^{-}.\top &\sqsubseteq \mathsf{JobPosition}\\
\exists\mathsf{suitableFor}.\top &\sqsubseteq \mathsf{PositivelyEvaluated} & \mathsf{EligibleUser} &\sqsubseteq \mathsf{User}\\
\mathsf{User} \sqcap \mathsf{Graduate} &\sqsubseteq \mathsf{EligibleUser} & \mathsf{EligibleUser} &\sqsubseteq \mathsf{Graduate}
\end{aligned}$$

while $\mathcal{A}$, which stores data on available job positions, contains the assertions

$$\begin{aligned}
&\mathsf{AcademicPosition}(\mathsf{professor}_{123}), & &\mathsf{AdminPosition}(\mathsf{secretary}_{123}),\\
&\mathsf{AcademicPosition}(\mathsf{researcher}_{123}), & &\mathsf{AdminPosition}(\mathsf{secretary}_{456}).
\end{aligned}$$

Moreover, we assume that $\mathcal{A}$ contains all the assertions of the form $\neg A(\mathsf{u})$, $\neg P(\mathsf{u}, a)$ and $\neg P(a, \mathsf{u})$, for a distinguished individual name $\mathsf{u} \in \mathsf{N_I}$ and every $A, P, a \in \Sigma_{\mathcal{O}}$, so that u can be used to represent an *undefined value*. The CIs of $\mathcal{T}$ formalise the following facts: there are both academic and administrative job positions and these are disjoint; users and job positions are disjoint; $\mathsf{appliesFor}$ relates users to job positions; to be suitable for something one has to be a user that is positively evaluated; the range of $\mathsf{suitableFor}$ is included in the extension of $\mathsf{JobPositions}$; an eligible user is defined as a graduate user. ◁

We define now the *standard translation* from $RDFS_{+}$ expressions to FO formulas, which maps concepts to FO formulas with one free variable, and roles to FO formulas with two free variables. Specifically, the translation T generates formulas that contain just two variables $x, y \in \mathsf{Var}$, and is defined as follows:

$$\begin{aligned}
T(A_1 \sqcap \cdots \sqcap A_n) &= A_1(x) \wedge \cdots \wedge A_n(x),\\
T(P) &= P(x, y), & T(P^{-}) &= P(y, x),\\
T(\exists R.\top) &= \exists y T(R), & T(\exists R.A) &= \exists y (T(R) \wedge A(y)),\\
T(\neg A) &= \neg T(A), & T(\neg R) &= \neg T(R),
\end{aligned}$$

where $A, A_1, \ldots, A_n$ are unary predicates and P is a binary predicate. Moreover, we map CIs and RIs into universal FO sentences in the following way:

$$T(C \sqsubseteq D) = \forall x (T(C) \rightarrow T(D)), \qquad T(R \sqsubseteq S) = \forall x \forall y (T(R) \rightarrow T(S)),$$

where D stands for either A or $\neg A$, and S stands for either R' or $\neg R'$. We also set $T(\mathcal{T}) = \bigcup_{\beta \in \mathcal{T}} \{T(\beta)\}$. Assertions α are (identically) mapped into FO

literals without free variables (i.e., *ground*), as $T(\alpha) = \alpha$, and we set $T(\mathcal{A}) = \bigcup_{\alpha \in \mathcal{A}} \{T(\alpha)\}$. Finally, $T(\mathcal{O}) = T(\mathcal{T}) \cup T(\mathcal{A})$. It is easy to see that the set of FO sentences obtained as the translation $T(\mathcal{O})$ of an $RDFS_+$ ontology $\mathcal{O}$, can be equivalently rewritten into a *universal Horn theory* [25,23]. Such a theory, which we identify with $T(\mathcal{O})$, can be obtained from $T(\mathcal{O})$ by simply putting formulas into prenex normal form.

The semantics for $RDFS_+$ expressions can be given in terms of their FO translation [25]. For an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and a concept C, we define the *extension* of C in $\mathcal{I}$ as $C^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \mathcal{I} \models T(C)[d]\}$. Similarly, for a role R, we define its extension in $\mathcal{I}$ as $R^{\mathcal{I}} = \{(d, e) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \mathcal{I} \models T(R)[d, e]\}$. We say that C and R are *satisfied* in $\mathcal{I}$ if $C^{\mathcal{I}} \neq \emptyset$ and $R^{\mathcal{I}} \neq \emptyset$, respectively.

Moreover, given a CI, RI, assertion, TBox, ABox, or ontology Γ, we say that Γ is *satisfied* in $\mathcal{I}$ (or that $\mathcal{I}$ is a *model* of Γ), written $\mathcal{I} \models \Gamma$, iff $\mathcal{I} \models T(\Gamma)$. Given an ontology $\mathcal{O}$ and (a concept, role, CI, RI, or assertion mapped, via its FO translation, into) an FO formula φ, we say that φ is *satisfiable w.r.t.* $\mathcal{O}$ (or $\mathcal{O}$*-satisfiable*) if there exists a model $\mathcal{I}$ of $\mathcal{O}$ that satisfies φ under some assignment in $\mathcal{I}$. Finally, we say that $\mathcal{O}$ *logically implies* an FO formula φ, or that φ is a *logical consequence* of $\mathcal{O}$, written $\mathcal{O} \models \varphi$, if, for every model $\mathcal{I}$ of $\mathcal{O}$ and every assignment $\mathfrak{a}$ in $\mathcal{I}$, we have that $\mathcal{I}$ satisfies φ under $\mathfrak{a}$.

3 Basic Model-Theoretic Properties

In this section, we prove the model-theoretic properties that will be used later on to develop our verification machinery. Specifically, we show here that the standard translation of the $RDFS_+$ ontologies introduced in the previous section admits model completion, and has the constraint satisfiability problem decidable. These properties will allow us, in the subsequent sections, to verify suitably defined DL-based data-aware processes by employing a variant of the SMT-based backward reachability procedure introduced in [10]. To present our results, we require the following preliminary notions.

A formula that is a conjunction of Σ-literals is called a Σ*-constraint*. Given a Σ-theory T, we define the *constraint satisfiability problem for* T as follows: given a formula $\exists \underline{y} \varphi(\underline{x}, \underline{y})$, where $\varphi(\underline{x}, \underline{y})$ is a Σ-constraint, decide whether $\exists \underline{y} \varphi(\underline{x}, \underline{y})$ is satisfiable w.r.t. T. A theory T has *quantifier elimination* iff, for every Σ_T-formula $\varphi(\underline{x})$, there exists a quantifier-free formula $\psi(\underline{x})$ such that $T \models \varphi(\underline{x}) \leftrightarrow \psi(\underline{x})$. Finally, we will use the following definition of model completion, which is restricted to cover the case of universal theories (the ones considered in this work) and that is nonetheless known to be equivalent (for universal theories) to the usual one from model theory [13,19,10]. Let T be a universal Σ-theory and let $T^* \supseteq T$ be a further Σ-theory. We say that T^* is a *model completion of* T iff: *(i)* every Σ-constraint satisfied in some model of T is also satisfied in some model of T^*; *(ii)* T^* has quantifier elimination.

We now state the main technical result of the section.

Theorem 2. *Given an RDFS$_+$ ontology $\mathcal{O}$, $T(\mathcal{O})$ is a finite universal FO theory that (i) has a decidable constraint satisfiability problem, and (ii) admits a model completion.*

Proof (Sketch). To prove Point *(i)*, we reduce to *RDFS*$_+$ (seen as a fragment of $\mathcal{ALCHI}$, [29]) ontology satisfiability. For Point *(ii)*, since there is no function symbol in $\Sigma_{T(\mathcal{O})}$, it is sufficient to show that $T(\mathcal{O})$ enjoys the amalgamation property: this is proved by explicitly constructing a $T(\mathcal{O})$-amalgam for every pair of models $\mathcal{I}_1$ and $\mathcal{I}_2$ of $T(\mathcal{O})$ sharing a submodel $\mathcal{I}_0$ (see [12] for details). □

Remark 3. For every *RDFS*$_+$ ontology $\mathcal{O}$, the model completion $T(\mathcal{O})^*$ of $T(\mathcal{O})$ admits quantifier elimination. The algorithm for quantifier elimination in $T(\mathcal{O})^*$ follows from the proof of Theorem 2: to eliminate $\exists x$ from a $\Sigma_{T(\mathcal{O})}$-formula $\exists x \varphi(x, \underline{y})$, it is sufficient to take the conjunction of the clauses $\chi(\underline{y})$ implied by $\varphi(x, \underline{y})$, which are finitely many for $T(\mathcal{O})$, up to $T(\mathcal{O})$-equivalence. This procedure is used in Algorithm 1 below and is crucial to get the decidability results of Theorem 6. ◁

Properties *(i)* and *(ii)* from Theorem 2 are in line with the foundational framework of [10,11], where a third property is additionally assumed: the *finite model property for constraint satisfiability* (see the references for the definition). However, differently from [10,11], this property is not needed anymore for the results of our paper. This is an important difference from [10,11], since the artifact systems studied there require to interact with *finite structures* (i.e., databases), whereas in the context of the present work we admit that the models of the knowledge base of our artifact systems can be infinite.

4 Ontology-Based Data-Aware Processes

In this section, we present our main contributions. We first define our model, called *RDFS*$_+$-based simple artifact systems, or *RDFS*$_+$-SASs for short, to formalise data-aware processes under *RDFS*$_+$ ontologies. These systems are a variant of the artifact-centric systems studied in [10]. *RDFS*$_+$-SASs read data from a given *RDFS*$_+$ ontology, used to store background information of the system, and manipulate individual variables, called *artifact variables*, which represent the current state of the process. We then study the parameterised safety problems of such models by adopting a symbolic version [21,11] of the well-known backward reachability procedure [1].

4.1 DL-Based Simple Artifact Systems

We first require the following preliminary notions. For an *RDFS*$_+$ ontology $\mathcal{O}$, an *$\mathcal{O}$-partition* is a finite set $P = \{\kappa_1(\underline{x}), \ldots, \kappa_n(\underline{x})\}$ of $\Sigma_{\mathcal{O}}$-literals such that $\mathcal{O} \models \forall \underline{x} \bigvee_{i=1}^{n} \kappa_i(\underline{x}) \wedge \forall \underline{x} \bigwedge_{i \neq j} \neg(\kappa_i(\underline{x}) \wedge \kappa_j(\underline{x}))$. Given an ontology $\mathcal{O}$, an $\mathcal{O}$-partition $P = \{\kappa_1(\underline{x}), \ldots, \kappa_n(\underline{x})\}$, and $\Sigma_{\mathcal{O}}$-terms $\underline{t}(\underline{x}) = (t_1(\underline{x}), \ldots, t_n(\underline{x}))$, (the value of) a *case-defined function F based on P and $\underline{t}$*, for a fresh function symbol

$F \in \mathsf{N_F}$, is defined as follows: for every model $\mathcal{I}$ of $\mathcal{O}$, every assignment $\mathfrak{a}$ in $\mathcal{I}$, and every tuple $\underline{x}$ of variables, $\mathfrak{a}(F(\underline{x})) = \mathfrak{a}(t_i(\underline{x}))$, if $\mathcal{I} \models^{\mathfrak{a}} \kappa_i(\underline{x})$.

In order to introduce verification problems in a symbolic setting, one first has to specify which formulas are used to represent *(i)* the sets of states, *(ii)* the system initialisations, and *(iii)* the system evolution. To capture these aspects, we provide the following definitions.

An *$RDFS_+$-based simple artifact system ($RDFS_+$-SAS)* is a tuple

$$\mathcal{S} = (\mathcal{O}, \underline{x}, \iota(\underline{x}), \bigcup_{j=1}^{m} \{\tau_j(\underline{x}, \underline{x}')\}),$$

where $m \in \mathbb{N}$, and

- $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is an $RDFS_+$ ontology;
- $\underline{x} = (x_1, \ldots, x_n)$ is a tuple of variables, called *artifact variables*, and $\underline{x}'$ is a tuple of variables that are renamed copies of variables in $\underline{x}$;
- $\iota(\underline{x}) = \bigwedge_{i=1}^{n} x_i = a_i$, with $a_i \in \mathsf{N_I}$, is an *initial state* formula;
- $\tau_j(\underline{x}, \underline{x}') = \exists \underline{y}(\gamma^j(\underline{x}, \underline{y}) \wedge \bigwedge_{i=1}^{n} x_i' = F_i^j(\underline{x}, \underline{y}))$, for $1 \leq j \leq m$, is a *transition formula*, where $\gamma^j(\underline{x}, \underline{y})$ is a conjunction of $\Sigma_{\mathcal{O}}$-literals called *guard* of τ_j, and $x_i' = F_i^j(\underline{x}, \underline{y})$, where each F_i^j is a case-defined function based on some $\mathcal{O}$-partition and list of $\Sigma_{\mathcal{O}}$-terms, is an *update* of τ_j.

Given an $RDFS_+$ ontology $\mathcal{O}$, we call *state ($\Sigma_{\mathcal{O}}$-)formula* a quantifier-free $\Sigma_{\mathcal{O}}$-formula $\varphi(\underline{x})$. A state formula constrains the content of the artifact variables characterising the current states of the systems. Notice that a state formula can represent a (possibly infinite) set of states, because of the presence of (possibly infinitely many) different elements in a model of the ontology $\mathcal{O}$. A *safety formula* for $\mathcal{S}$ is a state $\Sigma_{\mathcal{O}}$-formula $\nu(\underline{x})$, used to describe the undesired states of the system. We say that $\mathcal{S}$ is *safe w.r.t.* $\nu(\underline{x})$ if there does not exist $k \geq 0$ and a formula of the form

$$\iota(\underline{x}^0) \wedge \tau_{j_0}(\underline{x}^0, \underline{x}^1) \wedge \cdots \wedge \tau_{j_{k-1}}(\underline{x}^{k-1}, \underline{x}^k) \wedge \nu(\underline{x}^k), \qquad (\star)$$

that is satisfiable w.r.t. $\mathcal{O}$, where $1 \leq j_0, \ldots, j_{k-1} \leq m$ and each $\underline{x}^h$, with $0 \leq h \leq k$, is a tuple of variables that are renamed copies of variables in $\underline{x}$. The *safety problem for* $\mathcal{S}$ is the following decision problem: given a safety formula $\nu(\underline{x})$ for $\mathcal{S}$, decide whether $\mathcal{S}$ is safe w.r.t. $\nu(\underline{x})$. This verification problem is *parametric* on the models of a fixed $RDFS_+$ ontology, since safety is assessed *irrespectively of the choice of such a model*. This implies that, when the system is safe, it is so for *every* execution of the process under *every* possible model (which in principle are infinitely many) of the given ontology.

Example 4. We develop a simplified job hiring process for university personnel based on the domain knowledge formalised in Example 1. Each application is created using a dedicated website portal, where users that are potentially interested in applying need to register in advance. When a registered user decides to apply, the data created by this single application do not have to be stored persistently and thus can be maintained just by using artifact variables (described below) that can interact with the knowledge base. All these variables

are initialised with an undefined value u. In the first transition of the system, an application is created by a registered user, which falls into the extension of the concept User: the information about this user is then stored in the artifact variable $x_{\mathsf{applicant}}$. At this point, the application website asks the user whether they hold a university degree: in case of an affermative answer, the website accepts the user as eligible, the information about the user is stored using $x_{\mathsf{applicant}}$ and the process can progress. Then, the user picks up a job position (assigned to x_{job}) and applies for it. The following steps of the process involve the evaluation of the application: for both academic and administrative positions, if the eligible candidate is suitable for the chosen position, they are declared winner (assigned to x_{winner}), otherwise they are declared loser (assigned to x_{loser}). To formalise this process, we define the $RDFS_{+}$-SAS $\mathcal{S} = (\mathcal{O}, \underline{x}, \iota(\underline{x}), \bigcup_{j=1}^{7}\{\tau_j(\underline{x}, \underline{x}')\})$ so that:

- the ontology $\mathcal{O}$ is the $RDFS_{+}$ ontology given in Example 1;
- the artifact variables are $\underline{x} = (x_{\mathsf{applicant}}, x_{\mathsf{job}}, x_{\mathsf{eligible}}, x_{\mathsf{winner}}, x_{\mathsf{loser}})$;
- the initial state formula is
  $\iota = (x_{\mathsf{applicant}} = \mathsf{u}) \wedge (x_{\mathsf{job}} = \mathsf{u}) \wedge (x_{\mathsf{eligible}} = \mathsf{u}) \wedge (x_{\mathsf{winner}} = \mathsf{u}) \wedge (x_{\mathsf{loser}} = \mathsf{u})$;
- the transition formulas are
  $\tau_1 = \exists y_1(\mathsf{User}(y_1) \wedge x'_{\mathsf{applicant}} = y_1)$,
  $\tau_2 = \mathsf{EligibleUser}(x_{\mathsf{applicant}}) \wedge x'_{\mathsf{eligible}} = x_{\mathsf{applicant}}$,
  $\tau_3 = \exists z_1(\mathsf{JobPosition}(z_1) \wedge \mathsf{appliesFor}(x_{\mathsf{eligible}}, z_1) \wedge x'_{\mathsf{job}} = z_1)$,
  $\tau_4 = \mathsf{AcademicPosition}(x_{\mathsf{job}}) \wedge \mathsf{suitableFor}(x_{\mathsf{eligible}}, x_{\mathsf{job}}) \wedge x'_{\mathsf{winner}} = x_{\mathsf{eligible}}$,
  $\tau_5 = \mathsf{AdminPosition}(x_{\mathsf{job}}) \wedge \mathsf{suitableFor}(x_{\mathsf{eligible}}, x_{\mathsf{job}}) \wedge x'_{\mathsf{winner}} = x_{\mathsf{eligible}}$,
  $\tau_6 = \mathsf{AcademicPosition}(x_{\mathsf{job}}) \wedge \neg\mathsf{suitableFor}(x_{\mathsf{eligible}}, x_{\mathsf{job}}) \wedge x'_{\mathsf{loser}} = x_{\mathsf{eligible}}$,
  $\tau_7 = \mathsf{AdminPosition}(x_{\mathsf{job}}) \wedge \neg\mathsf{suitableFor}(x_{\mathsf{eligible}}, x_{\mathsf{job}}) \wedge x'_{\mathsf{loser}} = x_{\mathsf{eligible}}$.

An undesired situation of the system is the one where an applicant registered user is declared winner even if they were not eligible. This situation is formally described by the following safety formula for $\mathcal{S}$:

$$\nu = \mathsf{User}(x_{\mathsf{winner}}) \wedge \neg\mathsf{EligibleUser}(x_{\mathsf{winner}}). \qquad \triangleleft$$

4.2 Backward Search for $RDFS_{+}$-SASs

Algorithm 1 shows the *SMT-based backward reachability procedure* (or *backward search*) for handling the safety problem for an $RDFS_{+}$-SAS $\mathcal{S}$. An integral part of the algorithm is to compute *symbolic* preimages (Line 5). The intuition behind the algorithm is to execute a loop where, starting from the undesired states of the system (described by the safety formula $\nu(\underline{x})$), the state space of the system is explored *backward*: in every iteration of the while loop (Line 2), the current set of states is *regressed* through transitions thanks to the preimage computation. For that purpose, for any $\tau(\underline{z}, \underline{z}')$ and $\phi(\underline{z})$ (where $\underline{z}'$ are renamed copies of $\underline{z}$), we define $\tau := \bigvee_{h=1}^{m} \tau_h$ and $Pre(\tau, \phi)$ as the formula $\exists \underline{z}'(\tau(\underline{z}, \underline{z}') \wedge \phi(\underline{z}'))$. Let $\phi(\underline{x})$ be a state formula, describing the state of the artifact variables $\underline{x}$. The *preimage* of the set of states described by the formula $\phi(\underline{x})$ is the set of states described by $Pre(\tau, \phi)$ (notice that, when $\tau = \bigvee \hat{\tau}$, then $Pre(\tau, \phi) = \bigvee Pre(\hat{\tau}, \phi)$). We recall that a state formula is a quantifier-free $\Sigma_{\mathcal{O}}$-formula. Unfortunately, because of

Algorithm 1: SMT-based backward reachability procedure

```
   Function BReach(ν)
1  |  φ ⟵ ν; B ⟵ ⊥;
2  |  while φ ∧ ¬B is T(O)-satisfiable do
3  |  |  if ι ∧ φ is T(O)-satisfiable then
   |  |  |  return (unsafe, unsafe trace of form (⋆));
4  |  |  B ⟵ φ ∨ B;
5  |  |  φ ⟵ Pre(τ, φ);
6  |  |  φ ⟵ QE(T(O)*, φ);
   |  return safe;
```

the presence of the existentially quantified variables $\underline{y}$ in τ, $Pre(\tau, \phi)$ is *not* a state formula, in general. As stated in [10,11], if the quantified variables were not *eliminated*, we would break the *regressability* of the procedure: indeed, the states reached by computing preimages, intuitively described by $Pre(\tau, \phi)$, need to be represented by a state formula ϕ' in the new iteration of the while loop. In addition, the increase of the number of variables due to the iteration of the preimage computation would affect the performance of the satisfiability tests described below, in case the loop is executed many times. In order to solve these issues, it is essential to introduce the subprocedure $\mathsf{QE}(T(\mathcal{O})^*, \phi)$ in Line 6.

$\mathsf{QE}(T(\mathcal{O})^*, \phi)$ in Line 6 is a subprocedure that implements the quantifier elimination algorithm of $T(\mathcal{O})^*$ and that converts the preimage $Pre(\tau, \phi)$ of a state formula ϕ into a state formula (equivalent to it modulo the axioms of $T(\mathcal{O})^*$), so as to guarantee the regressability of the procedure: this conversion is possible since $T(\mathcal{O})^*$ eliminates from τ_h the existentially quantified variables $\underline{y}$. Backward search computes iterated preimages of the safety formula ν, until a fixpoint is reached (in that case, $\mathcal{S}$ is *safe* w.r.t. ν) or until a set intersecting the initial states (i.e., satisfying ι) is found (in that case, $\mathcal{S}$ is *unsafe* w.r.t. ν). *Inclusion* (Line 2) and *disjointness* (Line 3) tests can be discharged via proof obligations to be handled by SMT solvers. The fixpoint is reached when the test in Line 2 returns *unsat*: the preimage of the set of the current states is included in the set of states reached by the backward search so far (represented as the iterated application of preimages to the safety formula ν). The test at Line 3 is satisfiable when the states visited so far by the backward search includes a possible initial state (i.e., a state satisfying ι). If this is the case, then $\mathcal{S}$ is unsafe w.r.t. ν. Together with the unsafe outcome, the algorithm also returns an unsafe trace of the form $(\star)$, explicitly witnessing the sequence of transitions τ_h that, starting from the initial configurations, lead the system to a set of states satisfying the undesired conditions described by $\nu(\underline{x})$.

Theorem 5. *Backward search (Algorithm 1) is correct for detecting whether an $RDFS_+$-SAS $\mathcal{S}$ is safe w.r.t. $\nu(\underline{x})$.*

Proof (Sketch). First, we require the following claim, which follows immediately from the definitions.

Claim 1. For every safety formula $\nu(\underline{x})$ for $\mathcal{S}$ and every $k \geq 0$, a formula ϑ of the form $(\star)$ is satisfiable w.r.t. $\mathcal{O}$ iff ϑ is satisfiable w.r.t. $T(\mathcal{O})$.

Then, we need to show that, instead of considering satisfiability of formulas of the form $(\star)$ in models of $T(\mathcal{O})$, we can concentrate on satisfiability w.r.t. $T(\mathcal{O})^*$ ($T(\mathcal{O})^*$ exists thanks to Property *(ii)* of Theorem 2). Then, by exploiting the algorithm for quantifier elimination in $T(\mathcal{O})^*$ described in Remark 3, formulas of the form $(\star)$ can be represented via backward search by using quantifier-free formulas. We finally conclude by noticing that safety/unsafety of $\mathcal{S}$ w.r.t $\nu(\underline{x})$ can be now detected invoking the satisfiability tests (which are effective thanks to Property *(i)* of Theorem 2) over those quantifier-free formulae. □

Backward search for generic artifact systems is not guaranteed to terminate [11]. However, in case $\mathcal{S}$ is *unsafe* w.r.t. $\nu(\underline{x})$, an unsafe trace—which is finite—is found after finitely many iterations of the while loop: hence, in the unsafe case, backward search must terminate. Together with the theorem above, this means that the backward reachability procedure is at least a semi-decision procedure for detecting unsafety of RDFS_+-SASs. Nevertheless, we show in the following theorem that, in case of RDFS_+-SASs, backward search *always terminates*: thus, it is a full decision procedure, for which we also provide a PSPACE upper bound.

Theorem 6. *For an RDFS_+ ontology $\mathcal{O}$ and an RDFS_+-SAS $\mathcal{S} = (\mathcal{O}, \underline{x}, \iota(\underline{x}), \bigcup_{j=1}^{m}\{\tau_j(\underline{x}, \underline{x}')\})$, the safety problem for $\mathcal{S}$ is decidable in* PSPACE *in the combined size of $\underline{x}$, $\iota(\underline{x})$ and $\bigcup_{j=1}^{m}\{\tau_j(\underline{x}, \underline{x}')\}$.*

Proof (Sketch). For every RDFS_+ ontology $\mathcal{O}$, there are only finitely many quantifier-free $\Sigma_{T(\mathcal{O})}$-formulas, up to $T(\mathcal{O})$-equivalence, that can be built out of a finite set of variables $\underline{x}$. Thanks to the availability of the quantifier elimination procedure $\mathsf{QE}(T(\mathcal{O})^*, \varphi)$, the overall number of variables in φ is never increased. This implies that globally there are only finitely many quantifier-free $\Sigma_{T(\mathcal{O})}$-formulas that Algorithm 1 needs to analyse. Hence, Algorithm 1 terminates. Concerning complexity, we first note that the translation $T(\mathcal{O})$ requires polynomial time. Then, we need to eliminate the occurrences of case-defined functions (creating an equivalent SAS whose size is polynomial in the size of the original one), and to modify Algorithm 1 by making it nondeterministic with an NPSPACE complexity. The claim follows by applying Savitch's Theorem. □

We observe that Algorithm 1 is not yet implemented in the state-of-the-art model checker MCMT (Model Checker Modulo Theories [21]), which is based on SMT solving. Such an implementation, however, can be obtained by extending MCMT with the quantifier elimination algorithm for $T(\mathcal{O})^*$ (described in Remark 3), required in Line 6, together with a procedure for RDFS_+ ontology satisfiability (seen as a fragment of $\mathcal{ALCHI}$, [29]), required in Lines 2 and 3.

5 Conclusions

We have studied the problem of verification of data-aware processes under $RDFS_+$ ontologies, where the process component can interact with a knowledge base specified by means of the DL $RDFS_+$, underpinning the RDFS constructs. We addressed this problem by introducing a suitable model of DL-based artifact-centric systems, called $RDFS_+$-based SASs, and by leveraging the SMT-based version of the backward reachability procedure, which is a well-known technique to employ for verifying systems of this kind. Specifically, we showed that this procedure is a full decision procedure for detecting safety of $RDFS_+$-based SASs, and we also provided a PSPACE complexity upper bound.

This work opens several directions for future work. First, we notice that the choice of $RDFS_+$ ontologies is not intrinsic to our approach. Indeed, motivated by conceptual modelling and data integration issues in OBDA applications, we are currently working on the *DL-Lite family* of DLs, to define suitable *DL-Lite*-based SASs with analogous decidability and complexity results. The main difference we have to account for is that, for a *DL-Lite* ontology $\mathcal{O}$, we have an *equisatisfiable* (but not equivalent) translation into a universal one-variable FO sentence $T(\mathcal{O})$, and Claim 1 in the proof of Theorem 5 has to be modified to show that a trace ϑ is satisfiable w.r.t. $\mathcal{O}$ iff a suitably translated trace $\hat{\vartheta}$ is satisfiable w.r.t. $T(\mathcal{O})$. In general, nonetheless, we point out that *any* DL satisfying the two conditions stated in Theorem 2 can be chosen for our purposes, and that the same theoretical guarantees can be obtained over the SMT-based backward reachability procedure. As future work, we thus intend to introduce a more general framework for DL-based SASs that is able to account for different DLs. We also intend to extend the results obtained here to more sophisticated artifact-centric models, such as the *relational artifact systems* (*RASs*) studied in [10,11]. Moreover, it could be worth investigating in this setting also properties that go beyond safety, such as liveness and fairness.

Acknowledgements

This research has been partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, by the Italian Basic Research (PRIN) project HOPE, by the EU H2020 project INODE (grant agreement 863410) by the CHIST-ERA project PACMEL, by the project IDEE (FESR1133) funded by the Eur. Reg. Development Fund (ERDF) Investment for Growth and Jobs Programme 2014-2020, and by the Free University of Bozen-Bolzano through the projects KGID, GeoVKG, STyLoLa, VERBA and MENS.

References

1. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.: General decidability theorems for infinite-state systems. In: Proc. of the 11th Int. Conf. on Logic in Computer Science (LICS). pp. 313–321. IEEE Computer Society (1996). https://doi.org/10.1109/LICS.1996.561359
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. of Artificial Intelligence Research **36**, 1–69 (2009). https://doi.org/10.1613/jair.2820
3. Bagheri Hariri, B., Calvanese, D., De Giacomo, G., De Masellis, R., Montali, M., Felli, P.: Verification of description logic Knowledge and Action Bases. In: Proc. of the 20th Eur. Conf. on Artificial Intelligence (ECAI). pp. 103–108 (2012)
4. Bagheri Hariri, B., Calvanese, D., De Giacomo, G., Deutsch, A., Montali, M.: Verification of relational data-centric dynamic systems with external services. In: Proc. of the 32nd ACM Symp. on Principles of Database Systems (PODS). pp. 163–174 (2013)
5. Bagheri Hariri, B., Calvanese, D., Montali, M., De Giacomo, G., De Masellis, R., Felli, P.: Description logic Knowledge and Action Bases. J. of Artificial Intelligence Research **46**, 651–686 (2013). https://doi.org/10.1613/jair.3826
6. Brickley, D., Guha, R.: RDF Schema 1.1. W3C Recommendation, World Wide Web Consortium (2014), available at https://www.w3.org/TR/rdf-schema/
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning **39**(3), 385–429 (2007). https://doi.org/10.1007/s10817-007-9078-x
8. Calvanese, D., De Giacomo, G., Montali, M.: Foundations of data aware process analysis: A database theory perspective. In: Proc. of the 32nd ACM Symp. on Principles of Database Systems (PODS). pp. 1–12 (2013). https://doi.org/10.1145/2463664.2467796
9. Calvanese, D., Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Formal modeling and SMT-based parameterized verification of data-aware BPMN. In: Proc. of the 17th Int. Conf. on Business Process Management (BPM). Lecture Notes in Computer Science, vol. 11675, pp. 157–175. Springer (2019). https://doi.org/10.1007/978-3-030-26619-6_12
10. Calvanese, D., Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: From model completeness to verification of data aware processes. In: Description Logic, Theory Combination, and All That – Essays Dedicated to Franz Baader on the Occasion of his 60th Birthday. Lecture Notes in Computer Science, vol. 11560, pp. 212–239. Springer (2019). https://doi.org/10.1007/978-3-030-22102-7_10
11. Calvanese, D., Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: SMT-based verification of data-aware processes: A model-theoretic approach. Mathematical Structures in Computer Science **30**(3), 271–313 (2020). https://doi.org/10.1017/S0960129520000067
12. Calvanese, D., Gianola, A., Mazzullo, A., Montali, M.: SMT-Based Safety Verification of Data-Aware Processes under Ontologies (Extended Version). Tech. Rep. arXiv:2108.12330, arXiv.org e-Print archive (2021), available at https://arxiv.org/abs/2108.12330
13. Chang, C.C., Keisler, H.J.: Model Theory. North-Holland Publ. Co., 3rd edn. (1990)

14. Claßen, J., Liebenberg, M., Lakemeyer, G., Zarrieß, B.: Exploring the boundaries of decidable verification of non-terminating golog programs. In: Proc. of the 28th AAAI Conf. on Artificial Intelligence (AAAI). pp. 1012–1019. AAAI Press (2014)
15. Damaggio, E., Deutsch, A., Vianu, V.: Artifact systems with data dependencies and arithmetic. ACM Trans. on Database Systems **37**(3), 22 (2012). https://doi.org/10.1145/2338626.2338628
16. 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). pp. 252–267 (2009)
17. Deutsch, A., Hull, R., Vianu, V.: Automatic verification of database-centric systems. SIGMOD Record **43**(3), 5–17 (2014). https://doi.org/10.1145/2694428.2694430
18. Deutsch, A., Li, Y., Vianu, V.: Verification of hierarchical artifact systems. ACM Trans. on Database Systems **44**(3), 12:1–12:68 (2019). https://doi.org/10.1145/3321487
19. Ghilardi, S.: Model-theoretic methods in combined constraint satisfiability. J. of Automated Reasoning **33**(3–4), 221–249 (2004)
20. Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Petri nets with parameterised data - modelling and verification. In: Proc. of the 18th Int. Conf. on Business Process Management (BPM). Lecture Notes in Computer Science, vol. 12168, pp. 55–74. Springer (2020). https://doi.org/10.1007/978-3-030-58666-9_4
21. Ghilardi, S., Ranise, S.: Backward reachability of array-based systems by SMT solving: Termination and invariant synthesis. Log. Methods Comput. Sci. **6**(4) (2010). https://doi.org/10.2168/LMCS-6(4:10)2010
22. Ghilardi, S., Ranise, S.: MCMT: A model checker modulo theories. In: Giesl, J., Hähnle, R. (eds.) Proc. of the 5th Int. Joint Conf. on Automated Reasoning (IJCAR). Lecture Notes in Computer Science, vol. 6173, pp. 22–29. Springer (2010). https://doi.org/10.1007/978-3-642-14203-1_3
23. Hodges, W.: Model Theory, Encyclopedia of Mathematics and its applications, vol. 42. Cambridge University Press (1993)
24. Hull, R.: Artifact-centric business process models: Brief survey of research results and challenges. In: Proc. of the 7th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE). Lecture Notes in Computer Science, vol. 5332, pp. 1152–1163. Springer (2008). https://doi.org/10.1007/978-3-540-88873-4_17
25. Kontchakov, R., Zakharyaschev, M.: An introduction to description logics and query rewriting. In: Reasoning Web: Reasoning on the Web in the Big Data Era – 10th Int. Summer School Tutorial Lectures (RW). Lecture Notes in Computer Science, vol. 8714, pp. 195–244. Springer (2014). https://doi.org/10.1007/978-3-319-10587-1_5
26. de Leoni, M., Felli, P., Montali, M.: Strategy synthesis for data-aware dynamic systems with multiple actors. In: Calvanese, D., Erdem, E., Thielscher, M. (eds.) Proc. of the 17th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR). pp. 315–325 (2020). https://doi.org/10.24963/kr.2020/32
27. Montali, M., Rivkin, A., Ritter, D.: Formalizing integration patterns with multimedia data. In: Proc. of the 24th Int. Conf. on Enterprise Distributed Object Computing (EDOC). pp. 67–76. IEEE (2020). https://doi.org/10.1109/EDOC49727.2020.00018
28. Reichert, M.: Process and data: Two sides of the same coin? In: Proc. of the On the Move Confederated Int. Conf. (OTM). Lecture Notes in Computer Science, vol. 7565, pp. 2–19. Springer (2012). https://doi.org/10.1007/978-3-642-33606-5_2

29. Rudolph, S.: Foundations of description logics. In: Reasoning Web: Semantic Technologies for the Web of Data – 7th Int. Summer School Tutorial Lectures (RW). Lecture Notes in Computer Science, vol. 6848, pp. 76–136. Springer (2011). https://doi.org/10.1007/978-3-642-23032-5_2
30. Vianu, V.: Automatic verification of database-driven systems: a new frontier. In: Proc. of the 12th Int. Conf. on Database Theory (ICDT). pp. 1–13 (2009). https://doi.org/10.1145/1514894.1514896
31. Zarrieß, B., Claßen, J.: Decidable verification of golog programs over non-local effect actions. In: Proc. of the 30th AAAI Conf. on Artificial Intelligence (AAAI). pp. 1109–1115. AAAI Press (2016)