SMT Safety Verification of Ontology-Based Processes

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

[1]Faculty of Engineering, Free University of Bozen-Bolzano, Italy,
[2]Computing Science Department, Umeå University, Sweden
{calvanese, gianola, mazzullo, montali}@inf.unibz.it

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. This approach requires a combination of model-theoretic notions and algorithmic techniques based on backward reachability. We introduce here Ontology-Based Processes, which are 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. We prove that when the DL is expressed in (a slight extension of) RDFS, it enjoys suitable model-theoretic properties, and that by relying on such DL we can define Ontology-Based Processes to which backward reachability can still be applied. Relying on these results we are able to show that in this novel setting, verification of safety properties is decidable in PSPACE.

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 (Reichert 2012; Calvanese et al. 2019a; Ghilardi et al. 2020) and data management (Vianu 2009; Calvanese, De Giacomo, and Montali 2013; Bagheri Hariri et al. 2013a; Deutsch, Hull, and Vianu 2014; Deutsch, Li, and Vianu 2019). 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, and actions evolving them through their lifecycle are defined (Hull 2008). Different formal models have been proposed to capture artifact systems and study their verification (Calvanese, De Giacomo, and Montali 2013). One of the most studied settings considers artifact systems as being composed of: *(i)* a *read-only database* storing background information about artifacts that does not change during the system evolution; *(ii)* a *working memory*, used to store data that can be modified in the course of the evolution; and *(iii) transitions* (also called *actions* or *services*) that query the read-only database and the working memory and use the retrieved data to update the working memory. Verification of such systems is 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 content of the read-only database, thus calling for a particular form of parameterised verification (Damaggio, Deutsch, and Vianu 2012; Deutsch, Li, and Vianu 2019; Calvanese et al. 2019b, 2020).

In this paper, we study for the first time *ontology-based processes*, i.e., semantically-enriched artifact systems where the read-only database is substituted by a description logic (DL) ontology (Baader et al. 2003), 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 (i.e., the extensional component of the ontology), another where verification is against all possible models of a fixed ABox. In this work, we adopt the latter, and thus verify whether the system enjoys desired properties irrespectively of how the information explicitly provided by the ABox is completed through the assertions in the TBox (i.e., the intensional component of the ontology).

More in detail, we consider an extensively studied model of such artifact systems, called *simple artifact system* (*SAS*) by Calvanese et al. (2020), where the artifact working memory consists of a fixed set of *artifact variables* (Deutsch et al. 2009; Damaggio, Deutsch, and Vianu 2012; Calvanese et al. 2020). 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 (Brickley and Guha 2014), a schema/ontology language for the Semantic Web formalized by the W3C, and where the transitions that update the working memory are expressed over the ontology signature. For this setting, we provide an SMT-based backward reachability procedure to decide safety, showing that the problem is in PSPACE. To apply this machinery, the underlying DL must enjoy suitable model-theoretic properties. In particular, to this end, we prove for the first time that this DL admits a *model completion* (Chang and Keisler 1990). Showing the existence of model completions is not trivial: it requires a careful algebraic analysis of the class of *all* models. At the same time, we give indications on which DL constructs break our verification machinery: indeed, model

completions may not exist in general, and we show that simple examples such as RL-ontologies do not admit a model completion. This gives also a technical justification for the choice of RDFS.

Detailed proofs are provided in an extended version of this article (Calvanese et al. 2021).

Related work. In spirit, our approach is reminiscent of previous works studying the verification of dynamic systems, like Golog programs, operating over a DL ontology, such as those by Claßen et al. (2014), and Zarrieß and Claßen (2016). In fact, both in their settings and ours, the dynamic system evolves each model of the ontology, and properties are verified 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 (Bagheri Hariri et al. 2013b). However, we differ from the work by Claßen et al. (2014) and Zarrieß and Claßen (2016) in that our goal is not only to derive foundational results, but also to transfer them into practical algorithms and thus obtain a model that is readily implementable by relying on a state-of-the-art SMT-based model checker such as MCMT (Ghilardi and Ranise 2010). As customary for artifact 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 (de Leoni, Felli, and Montali 2020) or Petri nets interpreted with interleaving semantics (Ghilardi et al. 2020).

There exist very few approaches that have been proved to be successful for assessing parameterized safety of processes enriched with data capabilities, and SMT solvers are well-known to represent the state-of-the-art in this area (Calvanese et al. 2019a; Ghilardi et al. 2021; Calvanese et al. 2020): in this work we build in particular on top of the SMT-based verification framework from (Calvanese et al. 2020). A key element of novelty in our work is to lift, in conceptual and modelling terms, this SMT-based verification approach from to handle data stored in a DL ontology with incomplete information, as opposed to a standard relational database adopted in all the prior works on parameterized verification of artifact systems (Deutsch, Li, and Vianu 2019, 2016; Calvanese et al. 2020). It is well known that lifting results from settings with complete to ones with incomplete information is often a non-trivial task that requires novel insights, even more when dealing with data/knowledge dynamics and formal verification (where overall results are sparse and fragmented). This lies at the core of KR in AI and is a key contribution of our work. Moreover, to develop our novel results we need to extend the technical machinery from (Calvanese et al. 2020) in a non-trivial way. Indeed, the central technical result is then to show that model completion holds in this novel setting (Theorem 2), which in turn stands at the core of the verification machinery we employ. In this light, Theorem 2 is a crucial, non-trivial result, as proving the existence of a model completion calls every time for a sophisticated semantical analysis of FO models of the theory of interest, that is significantly different when a different theory is studied.

Preliminaries

In this section, we recall the syntax and semantics of first-order logic (FO). We then define the syntax of the DL *RDFS*$_+$, which is a slight extension of RDFS (Brickley and Guha 2014). Its semantics is given by the standard translation, mapping *RDFS*$_+$ ontologies into equivalent sets of FO formulas.

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* '$\exists$'. The definitions of a (*FO*) *term* t (with $\underline{t}$ denoting a possibly empty tuple of terms), *formula* φ, *atom*, and *literal* are given as usual. Moreover, we adopt the standard abbreviations and conventions for the other Boolean operators and the *universal quantifier* '$\forall$'. 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 are adopted for terms. A *sentence* is defined as a formula without free variables, while we call *quantifier-free* a formula without any existential or universal quantifiers. A formula is *existential* if it has the form $\exists \underline{x} \varphi(\underline{x})$, and *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 (respectively, terms), the *signature of* Γ, denoted Σ_Γ, is the set of predicate, function, and individual symbols occurring in Γ. Given Σ, we say that Γ is a set of Σ-*formulas* (resp., Σ-*terms*) if $\Sigma_\Gamma = \Sigma$.

Given a signature Σ, an (*FO*) Σ-*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 $P \in \mathsf{N_P} \cap \Sigma$; $f^\mathcal{I} \colon (\Delta^\mathcal{I})^{\mathsf{ar}(f)} \longrightarrow \Delta^\mathcal{I}$, for every $f \in \mathsf{N_F} \cap \Sigma$; and $a^\mathcal{I} \in \Delta^\mathcal{I}$, for every $a \in \mathsf{N_I} \cap \Sigma$. For a set Γ of terms or formulas, when no confusion can arise, we often write 'interpretation' in place of 'Σ_Γ-interpretation'. An *assignment* in $\mathcal{I}$ is a function $\mathfrak{a} \colon \mathsf{Var} \longrightarrow \Delta^\mathcal{I}$. We define the *value* of a Σ-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 $t = a \in \mathsf{N_I} \cap \Sigma$; and $\mathfrak{a}(t) = f^\mathcal{I}(\mathfrak{a}(\underline{t}))$, if $t = f(\underline{t})$, where $f \in \mathsf{N_F} \cap \Sigma$ 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:

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

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 $T \models \varphi \leftrightarrow \psi$.

Description Logics Preliminaries

The DL we consider here is an extension of RDFS (Brickley and Guha 2014) 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 this DL *RDFS*$_+$.

Formally, 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, where concept names are 1-ary and role names are 2-ary predicate symbols, hence $\mathsf{N_C} \cup \mathsf{N_R} \subseteq \mathsf{N_P}$. In *RDFS*$_+$, *concepts* C and *roles* R, respectively denoting 1-ary and 2-ary predicates, are defined as follows:

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

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 (Calvanese et al. 2007; Artale et al. 2009). Indeed, while *DL-Lite* allows for the use of existential quantification $\exists R.\top$ on the right-hand side of CIs, this is 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 $\exists R.A$ on the left-hand side of CIs, while this is not possible in *DL-Lite*. As we will see later, differently from *DL-Lite*, the FO translation of an *RDFS*$_+$ ontology is a universal theory.

$$\begin{array}{rcl}
\mathsf{AcadPos} & \sqsubseteq & \mathsf{JobPos} \\
\mathsf{AdminPos} & \sqsubseteq & \mathsf{JobPos} \\
\exists \mathsf{applFor}.\top & \sqsubseteq & \mathsf{User} \\
\exists \mathsf{suitFor}.\top & \sqsubseteq & \mathsf{User} \\
\exists \mathsf{suitFor}.\top & \sqsubseteq & \mathsf{GoodEval} \\
\mathsf{User} \sqcap \mathsf{Grad} & \sqsubseteq & \mathsf{EligUser}
\end{array}$$

$$\begin{array}{rcl}
\mathsf{AcadPos} & \sqsubseteq & \neg\mathsf{AdminPos} \\
\mathsf{User} & \sqsubseteq & \neg\mathsf{JobPos} \\
\exists \mathsf{applFor}^-.\top & \sqsubseteq & \mathsf{JobPos} \\
\exists \mathsf{suitFor}^-.\top & \sqsubseteq & \mathsf{JobPos} \\
\mathsf{EligUser} & \sqsubseteq & \mathsf{User} \\
\mathsf{EligUser} & \sqsubseteq & \mathsf{Grad}
\end{array}$$

$$\mathsf{AcadPos}(\mathsf{professor}_{123}), \quad \mathsf{AcadPos}(\mathsf{researcher}_{123}), \quad \mathsf{AdminPos}(\mathsf{secretary}_{123}), \quad \mathsf{AdminPos}(\mathsf{secretary}_{456}).$$

Figure 1: University personnel ontology for job hiring processes

Example 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 $\mathcal{T}$ and $\mathcal{A}$ contain the CIs and assertions shown in Figure 1. Moreover, we assume that $\mathcal{A}$, which stores data on available job positions, 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; applFor relates users to job positions; to be suitable for something one has to be a user that is positively evaluated; the range of suitFor is included in the extension of JobPos; 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 $x, y \in \mathsf{Var}$:

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

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:

$$\begin{array}{rcl}
T(C \sqsubseteq D) & = & \forall x(T(C) \rightarrow T(D)), \\
T(R \sqsubseteq S) & = & \forall x \forall y(T(R) \rightarrow T(S)),
\end{array}$$

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 $T(\mathcal{A}) = \bigcup_{\alpha \in \mathcal{A}} \{T(\alpha)\} = \mathcal{A}$. 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* (Kontchakov

and Zakharyaschev 2014; Hodges 1993). 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 (Kontchakov and Zakharyaschev 2014). For a concept C and an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, the *extension* of C in $\mathcal{I}$ is $C^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \mathcal{I} \models T(C)[d]\}$. Similarly, for a role R, its extension in $\mathcal{I}$ is $R^{\mathcal{I}} = \{(d, e) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \mathcal{I} \models T(R)[d, e]\}$. 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 $\mathcal{I} \models \Gamma$, if $\mathcal{I} \models T(\Gamma)$. 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 $\mathcal{I}$, $\mathcal{I}$ satisfies φ under $\mathfrak{a}$.

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 then allow us to verify suitably defined ontology-based processes by employing a variant of the SMT-based backward reachability procedure by Calvanese et al. (2020). For this, we require some 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* (QE) 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 (Chang and Keisler 1990; Ghilardi 2004). 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 a model of T is also satisfied in a model of T^*; *(ii)* T^* has QE. A model completion T^* of a theory T, when it exists, is unique: this justifies the use of the notation T^* for denoting the unique model completion of T.

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) For Point *(i)*, we reduce to *RDFS*$_+$ ontology satisfiability, which is decidable (since *RDFS*$_+$ is a fragment of $\mathcal{SROIQ}$ (Baader et al. 2017)). For Point *(ii)*, we require the following definitions. A theory T has the *amalgamation property* if, for every pair of embeddings $\mu_1 \colon \mathcal{I}_0 \longrightarrow \mathcal{I}_1$, $\mu_2 \colon \mathcal{I}_0 \longrightarrow \mathcal{I}_2$ between models $\mathcal{I}_0$ and $\mathcal{I}_1$, $\mathcal{I}_2$ of T, there exist a model $\mathcal{I}$ of T and embeddings $\nu_1 \colon \mathcal{I}_1 \longrightarrow \mathcal{I}$, $\nu_2 \colon \mathcal{I}_2 \longrightarrow \mathcal{I}$, such that $\nu_1 \circ \mu_1 = \nu_2 \circ \mu_2$ (we adopt here the usual notion of embedding (Chang and Keisler 1990)). The triple $(\mathcal{I}, \nu_1, \nu_2)$ (or, abusing notation, just $\mathcal{I}$) is called a T*-amalgam* of $\mathcal{I}_1, \mathcal{I}_2$ over $\mathcal{I}_0$. One can assume w.l.o.g. that $\mathcal{I}_0$ is a *substructure* of $\mathcal{I}_1$ and $\mathcal{I}_2$ and that μ_1 and μ_2 are inclusion embeddings (mapping an element of $\mathcal{I}_0$ in the same element of $\mathcal{I}_1$ and $\mathcal{I}_2$, resp.).

Since there is no function symbol in $\Sigma_{T(\mathcal{O})}$, it suffices to show that $T(\mathcal{O})$ enjoys the amalgamation property (Lipparini 1982). For this purpose, for every pair of models $\mathcal{I}_1$, $\mathcal{I}_2$ of $T(\mathcal{O})$ sharing a submodel $\mathcal{I}_0$, we define $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ as follows: *(i)* $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}_1} \cup \Delta^{\mathcal{I}_2}$; *(ii)* for every individual symbol $a \in \Sigma_{T(\mathcal{O})}$, $a^{\mathcal{I}} = a^{\mathcal{I}_1}$; *(iii)* for every (1- or 2-ary) predicate symbol $P \in \Sigma_{T(\mathcal{O})}$, $P^{\mathcal{I}} = P^{\mathcal{I}_1} \cup P^{\mathcal{I}_2}$. Observe that $a^{\mathcal{I}} = a^{\mathcal{I}_1} = a^{\mathcal{I}_0} = a^{\mathcal{I}_2}$. Moreover, if $\underline{d} \in P^{\mathcal{I}}$, where P is n-ary, for $n \in \{1, 2\}$, then $\underline{d} \in (\Delta^{\mathcal{I}_1})^n$ and $\underline{d} \in P^{\mathcal{I}_1}$, or $\underline{d} \in (\Delta^{\mathcal{I}_2})^n$ and $\underline{d} \in P^{\mathcal{I}_2}$: this follows from the definition of $P^{\mathcal{I}} := P^{\mathcal{I}_1} \cup P^{\mathcal{I}_2}$. Clearly, given embeddings $\mu_1 \colon \mathcal{I}_0 \longrightarrow \mathcal{I}_1$, $\mu_2 \colon \mathcal{I}_0 \longrightarrow \mathcal{I}_2$, the (inclusion) embeddings $i_1 \colon \mathcal{I}_1 \longrightarrow \mathcal{I}$, $i_2 \colon \mathcal{I}_2 \longrightarrow \mathcal{I}$ are such that $i_1 \circ \mu_1 = i_2 \circ \mu_2$. We show that $\mathcal{I}$ is a model of $T(\mathcal{O})$. A formula φ of $T(\mathcal{O})$ has one of the following forms:

(1) $\forall x (A_1(x) \wedge \cdots \wedge A_n(x) \rightarrow D(x))$;
(2) $\forall x \forall y (R_1(x, y) \rightarrow D(x))$;
(3) $\forall x \forall y (R_1(x, y) \wedge A(y) \rightarrow D(x))$;
(4) $\forall x \forall y (R_1(x, y) \rightarrow R_2(x, y))$;
(5) $\forall x \forall y (R_1(x, y) \rightarrow \neg R_2(x, y))$;

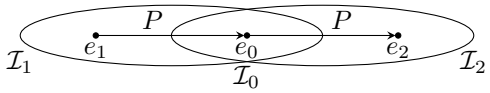
where: $A_k \in \mathsf{N_C}$, for $k \in \{1, \ldots, n\}$; $D \in \{B, \neg B\}$, with $B \in \mathsf{N_C}$; $R_i(x, y) = P_i(x, y)$, if $R_i = P_i$, and $R_i(x, y) = P_i(y, x)$, if $R_i = P_i^-$, with $P_i \in \mathsf{N_R}$ and $i \in \{1, 2\}$. Reasoning by cases, one can show that for every $j \in \{1, \ldots, 5\}$ and every formula $\varphi \in T(\mathcal{O})$ of the form (j), $\mathcal{I}$ is a model of φ. For example, we prove here Case (1). Given $d \in \Delta^{\mathcal{I}}$, suppose that $\mathcal{I} \models A_k[d]$, i.e., $d \in A_k^{\mathcal{I}}$, for all $k \in \{1, \ldots, n\}$. We have that $d \in \Delta^{\mathcal{I}_i}$ and $d \in A_k^{\mathcal{I}_i}$, for $i = 1$ or $i = 2$, and thus $\mathcal{I}_i \models A_k[d]$. Since $\mathcal{I}_i$ is a model of $T(\mathcal{O})$, and hence of φ, we have $\mathcal{I}_i \models D[d]$. Given that $D(x)$ is a literal and $\mathcal{I}_i$ is embedded in $\mathcal{I}$, we obtain that $\mathcal{I} \models D[d]$, and thus $\mathcal{I} \models \varphi$. This completes the proof that $T(\mathcal{O})$ has a model completion. □

Remark 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 $\exists x \varphi(x, \underline{y})$, we 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 and is crucial to get the decidability results of Theorem 9. ◁

Properties *(i)* and *(ii)* of Theorem 2 are in line with the foundational framework by Calvanese et al. (2020), where a third property is additionally assumed: the *finite model property for constraint satisfiability* (see the references for

the definition). However, an important difference with the work by Calvanese et al. (2020) is that this property here is not needed because we do not require *finite structures* (i.e., databases).

Finally, we observe that ontologies in the DL RL (Kontchakov and Zakharyaschev 2014) do not enjoy Property *(ii)* of Theorem 2. To see this, consider the RL ontology $\mathcal{O}_1 = (\mathcal{T}_1, \emptyset)$, with $\mathcal{T}_1 = \{\exists P.\exists P.\top \sqsubseteq \bot\}$. It can be shown (see figure below) that $T(\mathcal{O}_1)$ does not enjoy amalgamation, hence it does not admit a model completion (Chang and Keisler 1990).



Ontology-Based Processes

To study ontology-based processes under *RDFS*$_+$ ontologies, we introduce now our model, called *RDFS*$_+$-based processes, which are a variant of the artifact systems studied by Calvanese et al. (2019b). We also introduce the parameterised safety problems for our model, then studied in the following section.

Ontology-based processes 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.

To formalise such model, we first introduce case-defined functions. 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 $\mathcal{O}$-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 on P and $\underline{t}$*, for a fresh function symbol $F \in \mathsf{N_F}$, is 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 $\mathcal{I} \models^\mathfrak{a} \kappa_i(\underline{x})$. We call a case-defined function *trivial* when $i = 1$, that is, when it has only one case

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 process (RDFS*$_+$*-BP)* 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 each $x_i' = F_i^j(\underline{x}, \underline{y})$, where F_i^j is a case-defined function on some $\mathcal{O}$-partition and list of $\Sigma_\mathcal{O}$-terms, is called an *update* of τ_j.

Notice that, when the case-defined function F_i^j appearing in an update is trivial, the corresponding update $x_i' = F_i^j(\underline{x}, \underline{y})$ stands for a "value assignment" of the variable x_i' to a single term.

Example We develop a job hiring process for university personnel based on the domain knowledge from Example 1. Each application is created using a dedicated website portal, where potentially interested users need to register in advance. When a registered user decides to apply, the data created 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, an application is created by a registered user, which falls into the extension of the concept User: the information about the user is then stored in the artifact variable x_{appl}. At this point, the website asks the user whether they hold a university degree: if this is the case, the website accepts the user as eligible, their information is stored using x_{appl} 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 position, they are declared winner (assigned to x_{winr}), otherwise loser (assigned to x_{losr}). To formalise this process, we define the *RDFS*$_+$-BP $\mathcal{S} = (\mathcal{O}, \underline{x}, \iota(\underline{x}), \bigcup_{j=1}^7 \{\tau_j(\underline{x}, \underline{x}')\})$ so that $\mathcal{O}$ is the *RDFS*$_+$ ontology of Example 1, and

$\underline{x} = (x_{\mathsf{appl}}, x_{\mathsf{job}}, x_{\mathsf{elig}}, x_{\mathsf{winr}}, x_{\mathsf{losr}})$,
$\iota = \bigwedge_{x_i \in \underline{x}} x_i = \mathsf{u}$,
$\tau_1 = \exists y_1(\mathsf{User}(y_1) \wedge x'_{\mathsf{appl}} = y_1)$,
$\tau_2 = \mathsf{EligUser}(x_{\mathsf{applicant}}) \wedge x'_{\mathsf{elig}} = x_{\mathsf{appl}}$,
$\tau_3 = \exists z_1(\mathsf{JobPos}(z_1) \wedge \mathsf{applFor}(x_{\mathsf{elig}}, z_1) \wedge x'_{\mathsf{job}} = z_1)$;
$\tau_4 = \mathsf{AcadPos}(x_{\mathsf{job}}) \wedge \mathsf{suitFor}(x_{\mathsf{elig}}, x_{\mathsf{job}}) \wedge x'_{\mathsf{winr}} = x_{\mathsf{elig}}$,
$\tau_5 = \mathsf{AdminPos}(x_{\mathsf{job}}) \wedge \mathsf{suitFor}(x_{\mathsf{elig}}, x_{\mathsf{job}}) \wedge x'_{\mathsf{winr}} = x_{\mathsf{elig}}$,
$\tau_6 = \mathsf{AcadPos}(x_{\mathsf{job}}) \wedge \neg\mathsf{suitFor}(x_{\mathsf{elig}}, x_{\mathsf{job}}) \wedge x'_{\mathsf{losr}} = x_{\mathsf{elig}}$,
$\tau_7 = \mathsf{AdminPos}(x_{\mathsf{job}}) \wedge \neg\mathsf{suitFor}(x_{\mathsf{elig}}, x_{\mathsf{job}}) \wedge x'_{\mathsf{losr}} = x_{\mathsf{elig}}$. ◁

We define now parametric safety. 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 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* $\nu(\underline{x})$ for $\mathcal{S}$ is a state $\Sigma_\mathcal{O}$-formula describing the undesired states of the system. We say that $\mathcal{S}$ is *safe w.r.t.* $\nu(\underline{x})$ if there is no $k \geq 0$ and no formula

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

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

Algorithm 1: SMT-based backward reachability procedure

Function $\mathsf{BReach}(\nu)$

1 $\quad \phi \longleftarrow \nu$; $B \longleftarrow \bot$;
2 $\quad$ **while** $\phi \wedge \neg B$ *is* $T(\mathcal{O})$*-satisfiable* **do**
3 $\quad\quad$ **if** $\iota \wedge \phi$ *is* $T(\mathcal{O})$*-satisfiable* **then**
$\quad\quad\quad$ **return** (unsafe, *unsafe trace of form* $(\star)$);
4 $\quad\quad B \longleftarrow \phi \vee B$;
5 $\quad\quad \phi \longleftarrow \mathit{Pre}(\tau, \phi)$;
6 $\quad\quad \phi \longleftarrow \mathsf{QE}(T(\mathcal{O})^*, \phi)$;
$\quad$ **return** safe;

implies that, when the system is safe, it is so for *every* execution of the process under *every* model — in principle, infinitely many — of the given ontology.

Example Referring to Example 4, an undesired situation is the one where an applicant registered user is declared winner even if they were not eligible. This situation is formally described by the safety formula:

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

Verifying Safety Properties for *RDFS*$_+$-BPs

We study now the parameterised safety problems for *RDFS*$_+$-BPs by adopting a symbolic version (Calvanese et al. 2020) of the well-known backward reachability procedure (Abdulla et al. 1996). The requirements that are needed for employing our machinery are (cf. Theorem 2): (i) the existence of the model completion $T(\mathcal{O})^*$ for $T(\mathcal{O})$ (with an available quantifier elimination procedure in $T(\mathcal{O})^*$), and (ii) the decidability of the constraint satisfiability problem for the ontology $\mathcal{O}$.

Let us consider the safety problem for an *RDFS*$_+$-BP $\mathcal{S}$. First of all, we need to preprocess $\mathcal{S}$ in order to eliminate all the occurrences of case-defined functions. This can be done in polynomial time.

Lemma 6 *The safety problem for an RDFS*$_+$*-BP* $\mathcal{S}$ *can be reduced to the safety problem for an RDFS*$_+$*-BP* $\mathcal{S}'$ *(the size of which is polynomial in the size of* $\mathcal{S}$*) with only trivial case-defined functions.*

Indeed, similarly to what shown in (Calvanese et al. 2020), the previous lemma shows that also in this case case-defined functions can be eliminated w.l.o.g.From now on, we assume that *RDFS*$_+$-BPs $\mathcal{S}$ is without any case-defined function. We are ready to describe the main procedure for detecting safety of *RDFS*$_+$-BPs: this procedure will handle only the "pre-processed" RDFS+-BPs not containing case-defined functions

The *SMT-based backward reachability procedure* (or *backward search*) for handling the safety problem for an *RDFS*$_+$-BP $\mathcal{S}$ is shown in Algorithm 1. 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 $\mathit{Pre}(\tau, \phi)$ 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 $\mathit{Pre}(\tau, \phi)$ (notice that, when $\tau = \bigvee \hat{\tau}$, then $\mathit{Pre}(\tau, \phi) = \bigvee \mathit{Pre}(\hat{\tau}, \phi)$). We recall that a state formula is a quantifier-free $\Sigma_{\mathcal{O}}$-formula. Unfortunately, because of the presence of the existentially quantified variables $\underline{y}$ in τ, $\mathit{Pre}(\tau, \phi)$ is *not* a state formula, in general. As stated by Calvanese et al. (2020), 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 $\mathit{Pre}(\tau, \phi)$, need to be represented by a state formula ϕ' in the new iteration of the while loop. In addition, the increase in 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. $\mathsf{QE}(T(\mathcal{O})^*, \phi)$ implements the quantifier elimination algorithm of $T(\mathcal{O})^*$ and that converts the preimage $\mathit{Pre}(\tau, \phi)$ 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 include 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 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 7 *Backward search (Algorithm 1) is correct for detecting whether an RDFS*$_+$*-BP* $\mathcal{S}$ *is safe w.r.t.* $\nu(\underline{x})$.

Proof (Sketch) We first require the following claim, which comes immediately from the definitions.

Claim 8 *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.* $T(\mathcal{O})$.

We then need to show that, instead of satisfiability of formulas of the form $(\star)$ in models of $T(\mathcal{O})$, we can concentrate on satisfiability w.r.t. $T(\mathcal{O})^*$ (which 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 conclude by noticing that safety/unsafety of $\mathcal{S}$ w.r.t. $\nu(\underline{x})$ can be detected invoking the satisfiability tests on those quantifier-free formulas: these tests are effective thanks to Property *(i)* of Theorem 2, because the decidability of the constraint satisfiability problem implies the decidability of the satisfiability of arbitrary quantifier-free formulae. □

Backward search for generic artifact systems is not guaranteed to terminate (Calvanese et al. 2020). 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*$_+$-BPs. Nevertheless, we show in the following theorem that, in case of *RDFS*$_+$-BPs, backward search *always terminates*: thus, it is a full decision procedure, for which we also provide a PSPACE upper bound.

Theorem 9 *The safety problem for RDFS*$_+$*-BPs* $\mathcal{S} = (\mathcal{O}, \underline{x}, \iota(\underline{x}), \bigcup_{j=1}^{m}\{\tau_j(\underline{x}, \underline{x}')\})$ *is decidable in* PSPACE *in the combined size of* $\underline{x}$*,* ι*, and* $\bigcup_{j=1}^{m} \tau_j$*.*

Proof There are only finitely many quantifier-free $\Sigma_{T(\mathcal{O})}$-formulas, up to $T(\mathcal{O})$-equivalence, that could be built out of a finite set of variables $\underline{x}$: this holds for every *RDFS*$_+$ ontology $\mathcal{O}$. Thanks to the quantifier elimination procedure in Line 6, the overall number of variables in ϕ is never increased: notice that, without quantifier elimination, computing preimages $Pre(\tau, \phi_j)$ would introduce in ϕ_{j+1} new quantified variables, because of the presence of existentially quantified variables $\underline{y}$ in τ. 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 must terminate: by construction of B, the unsatisfiability test of Line 2 must eventually succeed, if the unsatisfiability test of Line 3 never does so.

Concerning complexity, we need to modify Algorithm 1. We first notice that, thanks to Lemma 6, the preprocessing that converts an *RDFS*$_+$-BP with occurrences of case-defined functions into its equivalent *RDFS*$_+$-BP without any occurrence of case-defined functions does not increase the overall complexity of the problem. Moreover, the translation of an *RDFS*$_+$-ontology $\mathcal{O}$ into $T(\mathcal{O})$ requires polynomial time. Since $T(\mathcal{O})$ is universal and without function symbols, also the satisfiability tests in Lines 2-3 can be performed in polynomial time (cf. (Calvanese et al. 2020), Proposition 3.1). Then, we adopt a nondeterministic procedure, analogous to the one by Calvanese et al. (2019b), Theorem 6.1, that makes the complexity NPSPACE: the main difference from (Calvanese et al. 2019b), Theorem 6.1, is that in our signatures, instead of unary functions, we have unary and binary relational symbols, but the argument works similarly. By Savitch's Theorem (PSPACE = NPSPACE), we conclude the proof. □

This upper bound is tight. In fact, propositional STRIPS planning (Bylander 1994), a well-known PSPACE-hard problem, can be directly encoded in our setting (without making use of an ontology, and only employing a propositional working memory).

We observe that Algorithm 1 is not yet implemented in the state-of-the-art model checker MCMT (Ghilardi and Ranise 2010), which is based on SMT solving. Such an implementation, however, can be obtained by extending MCMT with the quantifier elimination algorithm for $T(\mathcal{O})^*$ (cf. Remark 3), required in Line 6, together with a procedure for *RDFS*$_+$ ontology satisfiability, required in Lines 2–3.

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 ontology-based processes, called *RDFS*$_+$-BPs, 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 have shown that this procedure is a full decision procedure for detecting safety of *RDFS*$_+$-BPs, 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 Ontology-Based Data Access (Xiao et al. 2018) applications, we are currently working on the *DL-Lite family* of DLs, to define suitable *DL-Lite*-based processes 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 8 in the proof of Theorem 7 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 processes 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 by Calvanese et al. (2020).

Moreover, it is worth investigating in this setting also properties that go beyond safety, such as liveness. In this respect, we intend to exploit the framework proposed in (Geatti, Gianola, and Gigante 2022) for solving satisfiability of Linear Temporal Logic Modulo Theories over Finite Traces ($\mathsf{LTLf}^{\mathsf{MT}}$), which is a first-order extension of LTLf, so as to symbolically represent DL-based processes and express temporal properties over them. This approach is particularly promising because it relies on the use of the efficient BLACK solver (Geatti, Gigante, and Montanari 2019, 2021), which leverages SMT solvers as backend tools.

Acknowledgments

This research has been partially supported by the Italian Ministry of University and Research (MUR) under PRIN project PINPOINT Prot. 2020FNEB27, and by the Free University of Bozen-Bolzano with the ADAPTERS project.

References

Abdulla, P. A.; Cerans, K.; Jonsson, B.; and Tsay, Y.-K. 1996. General Decidability Theorems for Infinite-State Systems. In *Proc. of LICS*, 313–321.

Artale, A.; Calvanese, D.; Kontchakov, R.; and Zakharyaschev, M. 2009. The *DL-Lite* Family and Relations. *J. of Artificial Intelligence Research*, 36: 1–69.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge Univ. Press. ISBN 0-521-78176-0.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge Univ. Press. ISBN 978-0-521-69542-8.

Bagheri Hariri, B.; Calvanese, D.; De Giacomo, G.; Deutsch, A.; and Montali, M. 2013a. Verification of Relational Data-Centric Dynamic Systems with External Services. In *Proc. of PODS*, 163–174.

Bagheri Hariri, B.; Calvanese, D.; Montali, M.; De Giacomo, G.; De Masellis, R.; and Felli, P. 2013b. Description Logic Knowledge and Action Bases. *J. of Artificial Intelligence Research*, 46: 651–686.

Brickley, D.; and Guha, R. 2014. RDF Schema 1.1, W3C Recommendation, World Wide Web Consortium. https://www.w3.org/TR/rdf-schema/. Accessed: 2022-08-01.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artif. Intell.*, 69(1-2): 165–204.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. *J. Automat. Reason.*, 39(3): 385–429.

Calvanese, D.; De Giacomo, G.; and Montali, M. 2013. Foundations of Data Aware Process Analysis: A Database Theory Perspective. In *Proc. of PODS*, 1–12.

Calvanese, D.; Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2019a. Formal Modeling and SMT-Based Parameterized Verification of Data-Aware BPMN. In *Proc. of BPM*, volume 11675 of *LNCS*, 157–175. Springer.

Calvanese, D.; Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2019b. From Model Completeness to Verification of Data Aware Processes. In *Description Logic, Theory Combination, and All That*, volume 11560 of *LNCS*, 212–239. Springer.

Calvanese, D.; Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2020. SMT-based Verification of Data-Aware Processes: A Model-Theoretic Approach. *Math. Structures Comput. Sci.*, 30(3): 271–313.

Calvanese, D.; Gianola, A.; Mazzullo, A.; and Montali, M. 2021. SMT-Based Safety Verification of Data-Aware Processes under Ontologies (Extended Version). *CoRR*, abs/2108.12330.

Chang, C. C.; and Keisler, H. J. 1990. *Model Theory*. North-Holland Publ. Co.

Claßen, J.; Liebenberg, M.; Lakemeyer, G.; and Zarrieß, B. 2014. Exploring the Boundaries of Decidable Verification of Non-Terminating Golog Programs. In *Proc. of AAAI*, 1012–1019.

Damaggio, E.; Deutsch, A.; and Vianu, V. 2012. Artifact Systems with Data Dependencies and Arithmetic. *ACM Trans. Database Syst.*, 37(3): 22.

de Leoni, M.; Felli, P.; and Montali, M. 2020. Strategy Synthesis for Data-Aware Dynamic Systems with Multiple Actors. In *Proc. of KR*, 315–325.

Deutsch, A.; Hull, R.; Patrizi, F.; and Vianu, V. 2009. Automatic Verification of Data-Centric Business Processes. In *Proc. of ICDT*, 252–267.

Deutsch, A.; Hull, R.; and Vianu, V. 2014. Automatic Verification of Database-Centric Systems. *SIGMOD Record*, 43(3): 5–17.

Deutsch, A.; Li, Y.; and Vianu, V. 2016. Verification of Hierarchical Artifact Systems. In *Proceedings of PODS 2016*, 179–194. ACM.

Deutsch, A.; Li, Y.; and Vianu, V. 2019. Verification of Hierarchical Artifact Systems. *ACM Trans. Database Syst.*, 44(3): 12:1–12:68.

Geatti, L.; Gianola, A.; and Gigante, N. 2022. Linear Temporal Logic Modulo Theories over Finite Traces. In *Proc. of IJCAI 2022*, 2641–2647. ijcai.org.

Geatti, L.; Gigante, N.; and Montanari, A. 2019. A SAT-Based Encoding of the One-Pass and Tree-Shaped Tableau System for LTL. In *Proc. of TABLEAUX 2019*, volume 11714 of *LNCS*, 3–20. Springer.

Geatti, L.; Gigante, N.; and Montanari, A. 2021. BLACK: A Fast, Flexible and Reliable LTL Satisfiability Checker. In *Proc. of OVERLAY 2021, co-located with GandALF 2021*, volume 2987 of *CEUR Workshop Proceedings*, 7–12. CEUR-WS.org.

Ghilardi, S. 2004. Model-Theoretic Methods in Combined Constraint Satisfiability. *J. Automat. Reason.*, 33(3–4): 221–249.

Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2020. Petri Nets with Parameterised Data - Modelling and Verification. In *Proc. of BPM*, volume 12168 of *LNCS*, 55–74. Springer.

Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2021. Delta-BPMN: A Concrete Language and Verifier for Data-Aware BPMN. In *Proceedings of BPM 2021*, volume 12875 of *LNCS*, 179–196. Springer.

Ghilardi, S.; and Ranise, S. 2010. MCMT: A Model Checker Modulo Theories. In *Proc. of IJCAR*, volume 6173 of *LNCS*, 22–29. Springer.

Hodges, W. 1993. *Model Theory*. Cambridge Univ. Press.

Hull, R. 2008. Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In *Proc. of ODBASE*, volume 5332 of *LNCS*, 1152–1163. Springer.

Kontchakov, R.; and Zakharyaschev, M. 2014. An Introduction to Description Logics and Query Rewriting. In *Reasoning Web*, volume 8714 of *LNCS*, 195–244. Springer.

Lipparini, P. 1982. Locally Finite Theories with Model Companion. In *Atti Accad. Naz. Lincei*, volume 72.

Reichert, M. 2012. Process and Data: Two Sides of the Same Coin? In *Proc. of OTM*, volume 7565 of *LNCS*, 2–19. Springer.

Vianu, V. 2009. Automatic Verification of Database-Driven Systems: a New Frontier. In *Proc. of ICDT*, 1–13.

Xiao, G.; Calvanese, D.; Kontchakov, R.; Lembo, D.; Poggi, A.; Rosati, R.; and Zakharyaschev, M. 2018. Ontology-Based Data Access: A Survey. In Lang, J., ed., *Proc. of IJCAI 2018*, 5511–5519. ijcai.org.

Zarrieß, B.; and Claßen, J. 2016. Decidable Verification of Golog Programs over Non-Local Effect Actions. In *Proc. of AAAI*, 1109–1115.