Verification of Conjunctive-Query Based Semantic Artifacts*

Babak Bagheri Hariri[1], Diego Calvanese[1],
Giuseppe De Giacomo[2], and Riccardo De Masellis[2]

[1] KRDB Research Centre
Free University of Bozen-Bolzano
I-39100 Bolzano, Italy
{bagheri,calvanese}@inf.unibz.it

Dip. di Informatica e Sistemistica
Sapienza Università di Roma
Via Ariosto, 25, 00185 Rome, Italy
lastname@dis.uniroma1.it

Abstract. We introduce semantic artifacts, which are a mechanism that provides both a semantically rich representation of the information on the domain of interest in terms of an ontology, including the underlying data, and a set of actions to change such information over time. In this paper, the ontology is specified as a DL-Lite TBox together with an ABox that may contain both (known) constants and unknown individuals (labeled nulls, represented as Skolem terms). Actions are specified as sets of conditional effects, where conditions are based on conjunctive queries over the ontology (TBox and ABox), and effects are expressed in terms of new ABoxes. In this setting, which is obviously not finite state, we address the verification of temporal/dynamic properties expressed in μ-calculus. Notably, we show decidability of verification, under a suitable restriction inspired by the notion of acyclicity in data exchange.

1 Introduction

The artifact-centric approach to service modeling, introduced recently, considers both data and processes as first-class citizens in service design and analysis. Artifacts are advocated as a sort of middle ground between a conceptual formalization of a dynamic system and an actual implementation of the system itself [1]. The verification of temporal properties in the presence of data represents a significant challenge for the research community, since the system becomes infinite-state, and hence the usual analysis based on model checking of finite-state systems [2] is impossible in general. Recently, there have been some advancements on this issue, considering suitably constrained relational database settings for the data component (which acts also as a data storage for the process), see e.g., [3,4].

In this paper, we follow the line of [3], and rely on the work in knowledge representation to propose a more conceptual treatment of artifacts. We do so by enriching artifacts with a semantic layer constituted by a full-fledged ontology expressed in description logics. In particular, our *semantic artifacts* include an ontology specified in *DL-Lite*$_{\mathcal{R}}$ [5], which is the core of the web ontology language OWL2 QL[1], and it is particularly well suited for data management. The TBox of the ontology captures intensional information

* This work has been supported by the EU FP7-ICT Project ACSI (257593).

[1] http://www.w3.org/TR/owl2-profiles/

on the domain of interest, similarly to conceptual data models, such as UML class diagram, though as a software component to be used at runtime. The ABox, which acts as the artifact state, maintains the data of interest, which are accessed by relying on query answering through ontologies. As a query language, we use unions of conjunctive queries, possibly composing their certain answers through full FOL constructs [6]. A semantic artifact has associated actions whose execution changes the state of the artifact, i.e., its ABox. Such actions are specified as sets of conditional effects, where conditions are queries over the ontology and effects are expressed in terms of new ABoxes. To capture data that are acquired from external users/environments during the execution of actions, such ABoxes may contain special constants called labeled nulls, represented as *Skolem terms*. These represent individuals that the artifact does not know, but on which it knows some facts. Actions have no pre-condition, instead processes over the semantic artifact are used to specify which actions can be executed at each step. We model such processes as condition/action rules, where the condition is again expressed as a query over the ontology.

In this setting, which is obviously not finite state, we address the verification of temporal/dynamic properties expressed in μ-calculus [7], where atomic formulas are queries over the ontology that can refer only to known individuals. Unsurprisingly, we show that even for very simple semantic artifacts and dynamic properties verification is undecidable. However, we also show that for a very rich class of semantic artifacts, verification is indeed decidable and reducible to finite-state model checking. To obtain this result, we rely on recent results on the finiteness of the chase of tuple-generating dependencies in the data exchange literature [8].

2 Preliminaries

As ontology language, we make use of *DL-Lite*$_\mathcal{R}$, a member of the *DL-Lite* family [5], and hence a tractable DL particularly suited for dealing with ontologies (or KBs) with very large ABoxes, which can be managed through relational database technology. *DL-Lite*$_\mathcal{R}$ is also the core of the standard web ontology language OWL2 QL. In *DL-Lite*$_\mathcal{R}$, concepts and roles are formed according to the following syntax:

$$\begin{array}{lcl@{\qquad\qquad}lcl} B & ::= & N \mid \exists U & U & ::= & P \mid P^- \\ C & ::= & B \mid \neg B \mid \exists U.B & V & ::= & U \mid \neg U \end{array}$$

N, B, and C respectively denote a *concept name*, a *basic concept*, and an *arbitrary concept*. P, P^-, U, and V respectively denote a *role name*, an *inverse role*, a *basic role*, and an *arbitrary role*. A *DL-Lite*$_\mathcal{R}$ ontology is a pair (T, A), where T is a TBox, i.e., a finite set of (concept and role) *inclusion assertions* of the forms $B \sqsubseteq C$ and $U \sqsubseteq V$; and A is an ABox, i.e., a finite set of *facts* (also called membership assertions) of the forms $N(c_1)$ and $P(c_1, c_2)$, where N and P occur in T, and c_1 and c_2 are constants. We denote with $\mathcal{C}_A$ the set of constants appearing in A. The semantics of a *DL-Lite*$_\mathcal{R}$ ontology is the usual one for DLs, see [5]. Notice that in *DL-Lite*$_\mathcal{R}$ the unique name assumption is immaterial, since there is no way of deducing facts about equality. We say that A is *consistent wrt* T if (T, A) is satisfiable, i.e., admits at least one model.

A *union of conjunctive queries* (UCQ) q over a *DL-Lite*$_\mathcal{R}$ TBox T and constants $\mathcal{C}$ is a FOL formula of the form $\exists \boldsymbol{y}_1.conj_1(\boldsymbol{x}, \boldsymbol{y}_1) \vee \cdots \vee \exists \boldsymbol{y}_n.conj_n(\boldsymbol{x}, \boldsymbol{y}_n)$, with free

variables $\boldsymbol{x}$, existentially quantified variables $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$. Each $conj_i(\boldsymbol{x}, \boldsymbol{y_i})$ in q is a conjunction of atoms of the form $N(z)$, $P(z, z')$, $z = z'$, where N and P respectively denote a concept and a role name occurring in T, and z, z' are constants in $\mathcal{C}$ or variables in $\boldsymbol{x}$ or $\boldsymbol{y_i}$, for some $i \in \{1, \ldots, n\}$. Given constants $\mathcal{C}$ (typically $\mathcal{C}_A$), the *(certain-)answers to q over (T, A) wrt $\mathcal{C}$* is the set $ans_{\mathcal{C}}(q, T, A)$ of substitutions[2] θ of the free variables of q with constants in $\mathcal{C}$ such that $q\theta$ evaluates to true in every model of (T, A). We also consider an extension of UCQs, called ECQs, which are the range-restricted queries of the query language *EQL-Lite*(UCQ) [6], that is, the FOL query language whose atoms are UCQs evaluated according to the certain answer semantics above. Formally, an *ECQ* over T and $\mathcal{C}$ is a possibly open formula of the form

$$Q \ ::= \ q \mid Q_1 \wedge Q_2 \mid \neg Q \mid \exists x.Q,$$

where q denotes a UCQ over T and $\mathcal{C}$, with the proviso that every free variable of a negative subquery, i.e., of the form $\neg Q$, must occur in a positive subquery (to guarantee range-restriction). Given constants $\mathcal{C}$ the *answer to Q over (T, A) wrt $\mathcal{C}$*, is the set $ans_{\mathcal{C}}(Q, T, A)$ of tuples of constants in $\mathcal{C}$ obtained by *evaluating* the FOL formula Q in the standard way, while considering each UCQ q appearing in it as a primitive predicate with extension $ans_{\mathcal{C}}(q, T, A)$. For the connection with epistemic logic, see [6].

3 Semantic Artifacts

A *semantic artifact* $\mathcal{S} = (T, A_0, R)$ is a stateful device constituted by the information ontology (T, A_0) and the action specification R.

- T is a *DL-Lite*$_{\mathcal{R}}$ TBox, fixed once and for all, and capturing the intensional knowledge about the domain modeled by the semantic artifact.
- A_0 is a *DL-Lite*$_{\mathcal{R}}$ ABox, which expresses extensional information, and constitutes the *initial state* of the artifact. We call *proper constants* the constants $\mathcal{C}_{A_0}$ in A_0, and *labeled nulls* all new constants introduced by action execution.
- R is a set of *actions*, which change the state of the semantic artifact, i.e., the extensional information component.

An *action* ρ is constituted by a signature and an effect specification. The *action signature* is constituted by a name and a list of individual *input parameters*. Such parameters need to be substituted by constants for the execution of the action. Given a substitution θ for the input parameters, we denote by $\rho\theta$ the action with actual parameters.[3]

The *effect specification* consists of a set $\{e_1, \ldots, e_n\}$ of effects, which are assumed to take place simultaneously. Their formalization is inspired by the notion of mappings in data exchange [8]. Specifically, an *effect* e_i has the form $q_i^+ \wedge Q_i^- \rightsquigarrow A_i'$ where:

- $q_i^+ \wedge Q_i^-$ is a query over T and $\mathcal{C}_{A_0}$ with $\boldsymbol{x}$ as free variables, that may include some of the input parameters as query constants, q_i^+ is a UCQ, and Q_i^- is an arbitrary ECQ whose free variables are included in those of q_i^+, namely in $\boldsymbol{x}$. Intuitively, q_i^+ selects the tuples to instantiate the effect, and Q_i^- filters aways some of them.

[2] As customary, we can view each substitution simply as a tuple of constants, assuming some ordering of the free variables of q.

[3] We disregard a specific treatment of the output to the user, and assume instead that the user can freely pose queries to the ontology and thus extract implicit or explicit information from the states through which the semantic artifact evolves.

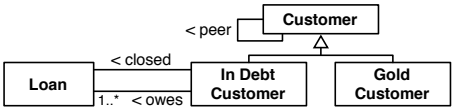

Fig. 1. A semantic artifact ontology

– A'_i is a set of facts over T, which include as constants: constants in $\mathcal{C}_{A_0}$, parameters, free variables of q_i^+, and implicitly existentially quantified variables.

Given a state A of $\mathcal{S}$, and a substitution σ for the parameters of the action ρ, the effect e_i extracts from A the set $ans_{\mathcal{C}_A}((q_i^+ \wedge Q_i^-)\sigma, T, A)$ of tuples of constants (proper constants and labeled nulls), and for each such tuple θ asserts a set $A'_i\sigma\theta$ of facts obtained from $A'_i\sigma$ by applying the substitution θ for the free variables of q_i^+. For each existentially quantified variable z in $A'_i\sigma$, a labeled null is introduced having the form $f_{z,e_i}(\boldsymbol{x})\sigma\theta$, where $f_{z,e_i}(\boldsymbol{x})$ is a Skolem function, depending on the existential variable z and the effect e_i, having as arguments the free variables $\boldsymbol{x}$ of q_i^+. We denote by $e_i\sigma(A)$ the overall set of facts, i.e., $e_i\sigma(A) = \bigcup_{\theta \in ans_{\mathcal{C}_A}(Q_i\sigma,T,A)} A'_i\sigma\theta$. The overall *effect* of the action ρ with parameter substitution σ over A is a new state $do(\rho\sigma, T, A) = \bigcup_{1\leq i\leq n} e_i\sigma(A)$ for $\mathcal{S}$. Notice that $do(\rho\sigma, T, A)$ may be inconsistent wrt T. In this case, the action ρ with σ over A is *not executable*.

Let's make some observations on such actions. The effects of an action are naturally a form of update of the previous state, and not of belief revision [9]. That is, we never learn new facts on the state in which an action is executed, but only on the state resulting from the action execution. We also observe that existentially quantified variables introduced by actions effects are used as witnesses of values chosen by the external user/environment when executing the action. We assume that such a choice depends only on the specific effects and the information retrieved by the query in the premises. We model such a choice by introducing suitable labeled nulls generated by appropriate Skolem functions. Finally, we observe that we do not make any persistence (or frame) assumption in our formalization [10]. In principle at every move we substitute the whole old state, i.e., ABox, with a new one. On the other hand, it should be clear that we can easily write effect specifications that *copy* big chunks of the old state into the new one. For example, $P(x, y) \rightsquigarrow P(x, y)$ copies the entire set of assertions involving the role P.

Example 1. Let us consider a semantic artifact $\mathcal{S} = (T, A_0, R)$, where T is the *DL-Lite*$_\mathcal{R}$ formalization of the UML diagram in Figure 1, which can be described as follows. A bank considers two specific types of customers: in-debt customers have a loan with the bank, while gold customers have access to special privileges. In-debt customers may have closed their loan. A relationship $\mathsf{peer}(c, p)$ between two customers denotes that p can vouch for c. The initial state is $A_0 = \{\mathsf{Gold}(\mathsf{john}), \mathsf{Cust}(\mathsf{ann}), \mathsf{peer}(\mathsf{mark}, \mathsf{john})\}$. R includes the following actions (we use brackets to isolate UCQs in ECQs):

$$\mathsf{GetLoan}(c) : \{\, [\exists p.\mathsf{peer}(c, p) \wedge \mathsf{Gold}(p)] \rightsquigarrow \{\mathsf{owes}(c, newl(c))\}, \quad CopyAll \;\}$$

That is, customer c gets a loan provided that s/he has a peer that is "gold". We use $CopyAll$ as a shortcut to denote effects that copy all concepts and roles.

$$\mathsf{CloseAllLoans}(c) : \{\, [\mathsf{owes}(c, l)] \rightsquigarrow \{\mathsf{closed}(c, l)\}, \quad CopyAll \;\}$$

That is, customer c closes all his/her loans which are moved to the closed relation.

$$\begin{aligned}\mathsf{UpdateDebts} : \{\ & [\mathsf{owes}(x,l)] \wedge [\neg\mathsf{closed}(x,l)] \rightsquigarrow \{\mathsf{owes}(x,l)\},\\ & [\mathsf{InDebt}(x)] \wedge [\forall l.\mathsf{owns}(x,l) \supset \mathsf{closed}(x,l)] \rightsquigarrow \{\mathsf{Cust}(x)\},\\ & \mathit{CopyAllExceptOwesClosedInDebt}\ \}\end{aligned}$$

That is, "remove" from owes those tuples that are in closed, and remove in-debt customers whose loans are all closed from InDebt, keeping them in Cust. $\mathit{CopyAllExceptOwes}$ copies everything except owes, closed, and InDebt. ■

4 Processes

Notice that semantic artifacts include information and actions to change such information. However, they do not say anything about how or when to apply a certain action. In other words, semantic artifact are agnostic to processes that use them. Processes over a semantic artifact $\mathcal{S} = (T, A_0, R)$ are (possibly nondeterministic) programs that use the state of $\mathcal{S}$ (accessed through T) to store their (intermediate and final) computation results, and the actions in R as atomic instructions. The state A can be arbitrarily queried through query answering over T, while it can be updated only through the actions in R. There are many ways to specify processes over $\mathcal{S}$. Here we adopt a rule-based specification.

A *condition/action rule* π for a semantic artifact $\mathcal{S}$ is an expression of the form $Q \mapsto \rho$, where ρ is an action in R and Q is an ECQ over T and $\mathcal{C}_{A_0}$, whose free variables are exactly the parameters of ρ. Such a rule expresses that, for each tuple θ for which condition Q holds, the action ρ with actual parameters θ *can* be executed.

A *process* is a finite set $\Pi = \{\pi_1, \ldots, \pi_n\}$ of rules. Notice that processes don't force the execution of actions but constrain them: the user of the process will be able to choose any of the actions that the rules forming the process allow. Notice also that our processes inherit entirely their states from the semantic artifact $\mathcal{S}$, see e.g., [1]. Other choices are also possible: the process could maintain its own state besides the one of the semantic artifact. As long as such an additional state is finite, or embeddable into the semantic artifact itself, the results here would easily extend to such a case.

The *execution* of a process Π over a semantic artifact $\mathcal{S}$ is defined as follows: we start from the initial state A_0 of the artifact, and for each rule $Q \mapsto \rho$ in Π, evaluate Q, and for each tuple θ returned execute $\rho\theta$, obtaining a new state $A' = do(\rho\theta, T, A_0)$ if A' is consistent wrt T (i.e., $\rho\theta$ is actually executable), and so on. In this way we build a *transition system* $\Upsilon(\Pi, \mathcal{S})$ whose states represent possible artifact states and where each transition represents the execution of an instantiated action that is allowed according to the process. $\Upsilon(\Pi, \mathcal{S})$ captures the behavior of the process Π over the artifact $\mathcal{S}$. In principle we can model-check such a transition system to verify dynamic properties. This is exactly what we are going to do next. However, one has to consider that in general such a transition system is infinite, so the classical results on model checking [2], which are developed for finite transition systems, do not apply.

Example 2. Referring to the example above, a process Π might include the following rules:

- $[\mathsf{Cust}(x)] \wedge \neg[\exists y.\mathsf{owes}(x,y)] \mapsto \mathsf{GetLoan}(x)$, i.e., a customer can get a loan if she does not have one already;
- $[\exists y.\mathsf{owes}(x,y)] \wedge [\exists y.(\mathsf{owes}(x,y) \wedge \neg\mathsf{closed}(x,y)] \mapsto \mathsf{CloseAllLoans}(x)$, i.e., a customer that owes loans that are not closed, can close them all at once;
- $\mathsf{true} \mapsto \mathsf{UpdateDebts}$, i.e., it is always possible to perform UpdateDebts. ■

5 Verification Formalism

We turn to the verification formalism to be used to specify dynamic properties of processes running over semantic artifacts. Several choices are possible. Here we focus on a variant of μ-calculus [7], which is one of the most powerful temporal logics that subsumes both linear time logics, such as LTL and PSL, and branching time logics such as CTL and CTL* [2]. In particular, we introduce a variant of μ-calculus, called $\mu\mathcal{L}$ that conforms with the basic assumption of our formalism, namely the use of ECQs to talk about the semantic information contained in the state of the artifact. Formally, given a semantic artifact $\mathcal{S} = (T, A_0, R)$, formulas of $\mu\mathcal{L}$ over $\mathcal{S}$ have the following form:

$$\Phi ::= Q \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \exists x.\Phi \mid \Box\Phi \mid \Diamond\Phi \mid \mu Z.\Phi \mid \nu Z.\Phi \mid Z$$

where Q is an ECQ over T and $\mathcal{C}_{A_0}$ (but not labeled nulls), and Z is a predicate variable.

The symbols μ and ν can be considered as quantifiers, and we make use of the notions of scope, bound and free occurrences of variables, closed formulas, etc. The definitions of these notions are the same as in first-order logic, treating μ and ν as quantifiers. In fact, we are interested only in closed formulas as specification of temporal properties to verify. For formulas of the form $\mu Z.\Phi$ and $\nu Z.\Phi$, we require the *syntactic monotonicity* of Φ wrt Z: every occurrence of the variable Z in Φ must be within the scope of an even number of negation signs. In μ-calculus, given the requirement of syntactic monotonicity, the least fixpoint $\mu Z.\Phi$ and the greatest fixpoint $\nu Z.\Phi$ always exist. In order to define the meaning of such formulas we resort to transition systems. Let $\mathfrak{A} = \Upsilon(\Pi, \mathcal{S})$ be the transition system for a process Π over a semantic artifact $\mathcal{S} = (T, A_0, R)$. We denote by $\Sigma_{\mathfrak{A}}$ the states of $\mathfrak{A}$. Let $\mathcal{V}$ be a predicate and individual variable valuation on $\mathfrak{A}$, i.e., a mapping from the predicate variables Z to subsets of the states $\Sigma_{\mathfrak{A}}$, and from individual variables to constants in $\mathcal{C}_{A_0}$. Then, we assign meaning to μ-calculus formulas by associating to $\Upsilon(\Pi, \mathcal{S})$ and $\mathcal{V}$ an *extension function* $(\cdot)^{\mathfrak{A}}_{\mathcal{V}}$, which maps μ-calculus formulas to subsets of $\Sigma_{\mathfrak{A}}$. The extension function $(\cdot)^{\mathfrak{A}}_{\mathcal{V}}$ is defined inductively as follows:

$$\begin{array}{ll}
(Q)^{\mathfrak{A}}_{\mathcal{V}} & = \{A \in \Sigma_{\mathfrak{A}} \mid ans_{\mathcal{C}_{A_0}}(Q\nu, T, A)\} \\
(Z)^{\mathfrak{A}}_{\mathcal{V}} & = Z\nu \subseteq \Sigma_{\mathfrak{A}} \\
(\neg\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \Sigma_{\mathfrak{A}} - (\Phi)^{\mathfrak{A}}_{\mathcal{V}} \\
(\Phi_1 \wedge \Phi_2)^{\mathfrak{A}}_{\mathcal{V}} & = (\Phi_1)^{\mathfrak{A}}_{\mathcal{V}} \cap (\Phi_2)^{\mathfrak{A}}_{\mathcal{V}} \\
(\exists x.\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \bigcup\{(\Phi)^{\mathfrak{A}}_{\mathcal{V}[x/c]} \mid c \in \mathcal{C}_{A_0}\} \\
(\Diamond\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \{A \in \Sigma_{\mathfrak{A}} \mid \exists A'.\ A \Rightarrow_{\mathfrak{A}} A' \text{ and } A' \in (\Phi)^{\mathfrak{A}}_{\mathcal{V}}\} \\
(\Box\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \{A \in \Sigma_{\mathfrak{A}} \mid \forall A'.\ A \Rightarrow_{\mathfrak{A}} A' \text{ implies } A' \in (\Phi)^{\mathfrak{A}}_{\mathcal{V}}\} \\
(\mu Z.\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \bigcap\{\mathcal{E} \subseteq \Sigma_{\mathfrak{A}} \mid (\Phi)^{\mathfrak{A}}_{\mathcal{V}[Z/\mathcal{E}]} \subseteq \mathcal{E}\} \\
(\nu Z.\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \bigcup\{\mathcal{E} \subseteq \Sigma_{\mathfrak{A}} \mid \mathcal{E} \subseteq (\Phi)^{\mathfrak{A}}_{\mathcal{V}[Z/\mathcal{E}]}\}
\end{array}$$

Here $A \Rightarrow_{\mathfrak{A}} A'$ holds iff there exists a rule $Q \mapsto \rho$ in Π such that there exists a $\theta \in ans_{\mathcal{C}_A}(Q, T, A)$ and $A' = do(\rho\theta, T, A')$. That is, there exist a rule in Π that can fire on A and produce an instantiated action $\rho\theta$, which applied on A has A' as effect.

Intuitively, the extension function $(\cdot)^{\mathfrak{A}}_{\mathcal{V}}$ assigns to the various μ-calculus constructs the following meanings. The boolean connectives have the expected meaning, while quantification is restricted to constants of $\mathcal{C}_{A_0}$, which are the only constants that the ECQs

in the formula can retrieve. We also use the usual FOL abbreviations. The extension of $\Diamond\Phi$ consists of the states A such that for *some* state A' with $A \Rightarrow_{\mathfrak{A}} A'$, we have that Φ holds in A'. While the extension of $\Box\Phi$ consists of the states A such that for *all* states A' with $A \Rightarrow_{\mathfrak{A}} A'$, we have that Φ holds in A'. The extension of $\mu Z.\Phi$ is the *smallest subset* $\mathcal{E}_\mu$ of $\Sigma_{\mathfrak{A}}$ such that, assigning to Z the extension $\mathcal{E}_\mu$, the resulting extension of Φ is contained in $\mathcal{E}_\mu$. That is, the extension of $\mu X.\Phi$ is the *least fixpoint* of the operator $\lambda\mathcal{E}.(\Phi)^{\mathfrak{A}}_{\mathcal{V}[Z/\mathcal{E}]}$ (here $\mathcal{V}[Z/\mathcal{E}]$ denotes the predicate valuation obtained from $\mathcal{V}$ by forcing the valuation of Z to be $\mathcal{E}$). Similarly, the extension of $\nu X.\Phi$ is the *greatest subset* $\mathcal{E}_\nu$ of $\Sigma_{\mathfrak{A}}$ such that, assigning to X the extension $\mathcal{E}_\nu$, the resulting extension of Φ contains $\mathcal{E}_\nu$. That is, the extension of $\nu X.\Phi$ is the *greatest fixpoint* of the operator $\lambda\mathcal{E}.(\Phi)^{\mathfrak{A}}_{\mathcal{V}[X/\mathcal{E}]}$. When Φ is a closed formula, $(\Phi)^{\mathfrak{A}}_{\mathcal{V}}$ does not depend on $\mathcal{V}$, and we denote it by $\Phi^{\mathfrak{A}}$.

The reasoning problem we are interested in is *model checking*, i.e., verify whether a $\mu\mathcal{L}$ *closed formula* Φ *holds for the process* Π *over the semantic artifact* $\mathcal{S}$. Formally, such a problem is defined as checking whether $A_0 \in \Phi^{\mathfrak{A}}$, that is, whether Φ is true in the root of the initial state A_0 of transition system $\Upsilon(\Pi, \mathcal{S})$.

Example 3. Continuing on our running example, we consider the following simple safety property: It is always true that gold customers in A_0 remain so. This property can be written as:

$$\forall x.([\mathsf{Gold}(x)] \supset \nu Z.([\mathsf{Gold}(x)] \wedge \Box Z)).$$

This formula is true, since no action (among the ones above) removes individuals from being Gold.

Next, we consider a simple liveness property: It is possible to reach a state in which a gold customer is also an in-debt customer.

$$\mu Z.([\exists x.\mathsf{Gold}(x) \wedge \mathsf{InDebt}(x)] \vee \Diamond Z)$$

This formula is true, because the ontology implies that if x participates to owes then x is an instance of InDebt; and we can reach a state in which $\exists x.\mathsf{Gold}(x) \wedge \mathsf{owes}(x, y)$ holds by firing the action $\mathsf{GetLoan}(\mathsf{john})$, which is allowed by the process. ■

6 Homomorphism and Bisimulation

We want to understand when two ABoxes A_1 and A_2 over a common *DL-Lite*$_{\mathcal{R}}$ TBox T provide the same answers to all EQL queries. Given two relational structures $\mathcal{I}_1$ and $\mathcal{I}_2$ over the same set of relations, and a set $\mathcal{C}$ of constants, a $\mathcal{C}$*-homomorphism* h from $\mathcal{I}_1$ to $\mathcal{I}_2$ is a mapping from the domain of $\mathcal{I}_1$ to the domain of $\mathcal{I}_2$ that preserves constants in $\mathcal{C}$ and relations, i.e., $h(c^{\mathcal{I}_1}) = c^{\mathcal{I}_2}$ for each $c \in \mathcal{C}$, and if $(d_1, \ldots, d_n) \in r^{\mathcal{I}_1}$, then $(h(d_1), \ldots, h(d_n)) \in r^{\mathcal{I}_2}$, for each relation r. Then, we say that there is a $\mathcal{C}$*-homomorphism from* A_1 *to* A_2 *wrt* T, denoted $A_1 \rightarrow^{\mathcal{C}}_T A_2$, iff there is a $\mathcal{C}$-homomorphism from $\mathcal{I}_{A_1}$ to each model of (T, A_2), where $\mathcal{I}_{A_1}$ is the structure whose domain is $\mathcal{C}_{A_1}$, whose constants are interpreted as themselves, and $N^{\mathcal{I}_{A_1}} = \{c \mid N(c) \in A_1\}$ for each concept name N, similarly for role names. This property can be checked by resorting to query answering over ontologies. For the ABox A_1, let Q_{A_1} be the boolean conjunctive query obtained as the conjunction of all facts in A_1, in which the constants not in $\mathcal{C}$ are treated as existentially quantified variables.

Lemma 1. *Given a DL-Lite*$_{\mathcal{R}}$ *TBox* T*, two ABoxes* A_1 *and* A_2 *over* T*, and a set* $\mathcal{C}$ *of constants, we have that* $A_1 \rightarrow^{\mathcal{C}}_T A_2$ *iff* $ans_{\mathcal{C}}(Q_{A_1}, T, A_2) = \mathsf{true}$.

Proof (sketch). We remind that a *DL-Lite*$_\mathcal{R}$ ontology (T, A) has a unique (up to renaming of domain elements) canonical-model [5], which is the model that has a $\mathcal{C}_A$-homomorphism to each model of (T, A). By composing homomorphisms, we have that $A_1 \rightarrow_T^{\mathcal{C}} A_2$ iff there is a $\mathcal{C}$-homomorphism from $\mathcal{I}_{A_1}$ to the canonical model of (T, A_2). The claim then follows by considering that there is a $\mathcal{C}$-homomorphism from $\mathcal{I}_{A_1}$ to the canonical model of (T, A_2) iff $ans_\mathcal{C}(Q_{A_1}, T, A_2) = \mathsf{true}$ [11,5]. □

A_1 and A_2 are said *$\mathcal{C}$-homomorphically equivalent wrt T* if $A_1 \rightarrow_T^{\mathcal{C}} A_2$ and $A_2 \rightarrow_T^{\mathcal{C}} A_1$.

Theorem 1. *Let $\mathcal{C}$ be a set of constants, and A_1, A_2 two ABoxes that are $\mathcal{C}$-homomorphically equivalent wrt a TBox T. Then, for every ECQ Q over T using only constants in $\mathcal{C}$, we have that $ans_\mathcal{C}(Q, T, A_1) = ans_\mathcal{C}(Q, T, A_2)$.*

Next we want to capture when two states of a single transition system or more generally of two transition systems, possibly obtained by applying different processes to different semantic artifacts sharing the same TBox and constants in the initial state, can be considered *behaviourally equivalent*, in the sense that they satisfy exactly the same $\mu\mathcal{L}$ formulas. To formally capture such an equivalence, we make use of the notion of *bisimulation* [12], suitably extended to deal with query answering over ontologies.

Given two artifact transition systems $\mathfrak{A}_1 = \Upsilon(\Pi_1, \mathcal{S}_1)$ (with states $\Sigma_{\mathfrak{A}_1}$) and $\mathfrak{A}_2 = \Upsilon(\Pi_2, \mathcal{S}_2)$ (with states $\Sigma_{\mathfrak{A}_2}$) such that $\mathcal{S}_1 = (T, A_{10}, R_1)$ and $\mathcal{S}_2 = (T, A_{20}, R_2)$ share the same TBox T and the same constants $\mathcal{C} = \mathcal{C}_{A_{10}} = \mathcal{C}_{A_{20}}$, a *bisimulation* is a relation $\mathcal{B} \subseteq \Sigma_{\mathfrak{A}_1} \times \Sigma_{\mathfrak{A}_2}$ such that: $(A_1, A_2) \in \mathcal{B}$ implies that:

1. A_1 and A_2 are $\mathcal{C}$-homomorphically equivalent wrt T;
2. if $A_1 \Rightarrow_{\mathfrak{A}_1} A_1'$ then there exists A_2' such that $A_2 \Rightarrow_{\mathfrak{A}_2} A_2'$ and $(A_1', A_2') \in \mathcal{B}$;
3. if $A_2 \Rightarrow_{\mathfrak{A}_2} A_2'$ then there exists A_1' such that $A_1 \Rightarrow_{\mathfrak{A}_1} A_1'$ and $(A_1', A_2') \in \mathcal{B}$.

We say that two states A_1 and A_2 are *bisimilar*, if there exists a bisimulation $\mathcal{B}$ such that $(A_1, A_2) \in \mathcal{B}$. Two transition systems $\mathfrak{A}_1$ with initial state A_{10} and $\mathfrak{A}_2$ with initial state A_{20} are *bisimilar* if $(A_{10}, A_{20}) \in \mathcal{B}$.

The following theorem states that the formula evaluation in $\mu\mathcal{L}$ is indeed invariant wrt bisimulation, so we can equivalently check any bisimilar transition systems.

Theorem 2. *Let $\mathfrak{A}_1$ and $\mathfrak{A}_2$ be two transition systems obtained from two semantic artifacts sharing the same TBox and constants. Then, for two states A_1 of $\mathfrak{A}_1$ and A_2 of $\mathfrak{A}_2$ (including the initial ones) are bisimilar iff for all $\mu\mathcal{L}$ closed formulas Φ over the two semantic artifacts, we have that $A_1 \in (\Phi)^{\mathfrak{A}_1}$ iff $A_2 \in (\Phi)^{\mathfrak{A}_2}$.*

Proof. The proof is analogous to the standard proof of bisimulation invariance of μ-calculus [7], though taking into account our specific definition of bisimulation, using Theorem 1 to guarantee that ECQs are evaluated identically over bisimilar states. □

7 Undecidability and Decidability

We now show that, not surprisingly, verification in the infinite state setting we considered is undecidable, but that it becomes decidable under some interesting conditions inspired by the recent literature on data exchange [8]. Our results rely on the possibility of building special semantic artifacts that we call "inflationary approximates". For such

special artifacts there exists a tight correspondence between the application of an action and a step in the chase of a set of tuple-generating dependencies (TGDs) [13,8]

Given a semantic artifact $\mathcal{S} = (T, A_0, R)$, its *inflationary approximate* is the semantic artifact $\mathcal{S}^+ = (T^+, A_0, R^+)$ defined as follows. T^+ is obtained from T by dropping all assertions involving negation on the right-hand side, thus obtaining a TBox formed by positive inclusions only. R^+ is formed by one action specification ρ^+ for each action specification $\rho \in R$, where ρ^+ is obtained from ρ by:

- removing all input parameters from the signature – note that the variables in q_i^+ that used to be parameters in ρ, become free variables in ρ^+;
- substituting each effect $e_i : q_i^+ \wedge Q_i^- \rightsquigarrow A_i'$ with $e_i : q_i^+ \rightsquigarrow A_i'$ – note that we need to preserve the Skolem functions name in the transformation;
- adding effects to copy all concept and role names, namely adding an effect $N(x) \rightsquigarrow N(x)$ for each concept name N of T, and an effect $P(x, y) \rightsquigarrow P(x, y)$ for each role name P of T.

Observe that executing actions in $\mathcal{S}^+$ can never give rise to an inconsistency, since T^+ does not contain any negative information [5].

We also consider the *generic process* $\Pi_\top$, in which all condition/action rules have the trivially true condition. Hence, $\Pi_\top$ allows for executing every action at every step. With these notions in place, it is easy to prove that verification in this setting is undecidable.

Theorem 3. *$\mu\mathcal{L}$ model checking of processes over semantic artifacts is undecidable.*

Proof (sketch). We show that it is already undecidable to check, given a semantic artifact $\mathcal{S}_\emptyset^+ = (\emptyset, A_0, R)$, of the form of an inflationary approximate of an artifact with an empty TBox, and the transition system $\mathfrak{A} = \Upsilon(\Pi_\top, \mathcal{S}_\emptyset^+)$, whether $A_0 \in \mu Z(q \vee \Diamond Z)^{\mathfrak{A}}$, where q is a boolean UCQ. We observe that the set of all actions in $\mathcal{S}^+$ can be seen as a set of TGDs, indeed it suffices to consider one TGD for each disjunct in the UCQ on the left-hand side of an effect specification. So, we can reduce to the above model checking problem the problem of answering boolean UCQs in a relational database under a set of TGDs, which is undecidable [14] □

Next, we observe a notable property of the transition system $\Upsilon(\Pi_\top, \mathcal{S}^+)$ generated by the generic process $\Pi_\top$ over the inflationary approximate $\mathcal{S}^+$ of a semantic artifact $\mathcal{S}$. Namely, each $do(\rho^+, T, \cdot)$ is a monotonic operator. This implies that by repeatedly applying such operators starting from the ABox A_0 in $\mathcal{S}^+$, we get at the limit (possibly transfinite) a single ABox A_{max}, which is the *least fixpoint* of such operators taken collectively [15,16]. Such an ABox contains, every fact generated by repeatedly executing actions from the inflationary approximate $\mathcal{S}^+$, that is every state A^+ of $\Upsilon(\Pi_\top, \mathcal{S}^+)$ is such that A^+ is contained in A_{max}. More interestingly, we show next that A_{max} contains also every A generated by repeatedly executing actions from the original $\mathcal{S}$.

Lemma 2. *Let $\mathcal{S} = (T, A_0, R)$ be a semantic artifact and Π a process over $\mathcal{S}$. Then every state A of the transition system $\Upsilon(\Pi, \mathcal{S})$ is a subset of A_{max} defined as above.*

Proof (sketch). We can show by induction that, for every sequence of actions $\rho_1\theta_1, \rho_2\theta_2, \ldots, \rho_n\theta_n$ generated by the process Π starting from A_0, the resulting state $do(\rho_n\theta_n, T, do(\cdots do(\rho_2\theta_2, T, do(\rho_1\theta_1, T, A_0)) \cdots))$ is a subset of the corresponding resulting state $do(\rho_n^+T^+, do(\cdots do(\rho_2^+, T^+, do(\rho_1^+, T^+, A_0)) \cdots))$ of the inflationary approx., which is a subset of A_{max}. □

In other words, A_{max} generated by the generic process $\Pi_\top$ running over the inflationary approximate $\mathcal{S}^+$ of a semantic artifact $\mathcal{S}$, bounds all states A that any process Π can generate by running on $\mathcal{S}$. Hence if for any reason A_{max} is finite, then the transition system $\Upsilon(\Pi, \mathcal{S})$ is *finite*. Hence, being model checking of finite transition system decidable (in fact polynomial in the size of the transition system), we get that also model checking of $\Upsilon(\Pi, \mathcal{S})$ is decidable.

To get finiteness guarantees on A_{max}, we take advantage of the correspondence between action execution and TGDs chase steps, as in the proof of Theorem 3. We build on this correspondence by further considering that in *DL-Lite*$_\mathcal{R}$, every UCQ q over a TBox can be rewritten as a new UCQ $rew(q)$ over the same alphabet, to be *evaluated* over the ABox considered as a relational database [5] (that is dropping the TBox).

In the literature for data exchange, several conditions that guarantee the finiteness of the chase of TGDs have been considered [17,18]. Here we focus on the original notion of *weakly-acyclic* TGDs [17]. Weak-acyclicity is a syntactic notion that involves the so-called *dependency graph* of the set of TGDs. Informally, a set D of TGDs is weakly-acyclic if there are no cycles in the dependency graph of D involving "existential" relation positions. The key property of *weakly-acyclic* TGDs is that chasing a relational database with them (i.e., applying them in all possible ways) generates a set of facts (a database) that is finite. See [17] for details.

Given a semantic artifact $\mathcal{S} = (T, A_0, R)$ and its inflationary approximate $\mathcal{S}^+ = (T^+, A_0, R^+)$, we define the set $E^+_\emptyset$ of effect specifications that includes an effect $rew(q^+_i) \rightsquigarrow A_i$ for each effect $q^+_i \rightsquigarrow A_i$ of an action $\rho^+ \in R^+$. Notice that the set $E^+_\emptyset$ can be seen as a set of TGDs. We say that a semantic artifact $\mathcal{S}$ is *weakly-acyclic* if the set $E^+_\emptyset$ seen as a set of TGDs is weakly-acyclic. (Note that the semantic artifact in our example is trivially weakly-acyclic.)

Lemma 3. *Let $\mathcal{S} = (T, A_0, R)$ be a weakly-acyclic semantic artifact and S^+ its inflationary approximate. Then A_{max} computed as above for S^+ is finite.*

Proof (sketch). We have to show that starting from A_0 we get to the least fixpoint A_{max} of the $do(\rho^+, T, \cdot)$ operator in a finite number of steps. To do so, we follow the line of the proof of finiteness of chase of weakly-acyclic TGDs in [17], and show that the number of Skolem terms generated by the effects of actions is bounded by a polynomial in the size of A_0. Differently from [17], we cannot rely on the notion of homomorphism to stop firing actions, but have to use membership of the new set of facts in the previous ones. □

As a direct consequence of Lemma 2 and Lemma 3, for weakly-acyclic semantic artifacts verification is decidable.

Theorem 4. *$\mu\mathcal{L}$ model checking of processes over weakly-acyclic semantic artifacts is decidable.*

Proof (sketch). The result follows by observing that every state generated by the execution of any process Π over a weakly-acyclic semantic artifact $\mathcal{S}$ is a subset of A_{max}, which by Lemma 3 is finite. Hence, we can apply a model checking procedure for μ-calculus on finite-state systems [7]. □

Note that the proof of Theorem 4 is giving us a single exponential upper bound (in the size of A_0) for $\mu\mathcal{L}$ model checking involving weakly-acyclic semantic artifacts.

8 Conclusions

In this paper we have studied verification of processes over semantic artifacts. We obtain an interesting decidability result by relying on the notion of inflationary approximate, which allows for a connection with the theory of chase of TGDs in relational databases. We close by observing that while we fully used the ontology for querying the artifact state, we use it in a limited way when *updating* the state, namely only to guarantee consistency. Ontology update has its own semantic and computational difficulties, see e.g., [19], which our approach sidesteps. However, if one could introduce a suitable notion of inflationary approximate in that case, the approach presented here could be used to devise decidable cases.

References

1. Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Bull. on Data Engineering **32**(3) (2009) 3–9
2. Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. The MIT Press (1999)
3. Cangialosi, P., De Giacomo, G., De Masellis, R., Rosati, R.: Conjunctive artifact-centric services. In: Proc. of ICSOC 2010. Volume 6470 of LNCS., Springer (2010) 318–333
4. Damaggio, E., Deutsch, A., Vianu, V.: Artifact systems with data dependencies and arithmetic. In: Proc. of ICDT 2011. (2011) 66–77
5. 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) (2007) 385–429
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: EQL-Lite: Effective first-order query processing in description logics. In: Proc. of IJCAI 2007. (2007) 274–279
7. Emerson, E.A.: Automated temporal reasoning about reactive systems. In: Logics for Concurrency: Structure versus Automata. Volume 1043 of LNCS. Springer (1996) 41–101
8. Kolaitis, P.G.: Schema mappings, data exchange, and metadata management. In: Proc. of PODS 2005. (2005) 61–75
9. Katsuno, H., Mendelzon, A.: On the difference between updating a knowledge base and revising it. In: Proc. of KR'91. (1991)
10. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. The MIT Press (2001)
11. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: Proc. of STOC'77. (1977) 77–90
12. Milner, R.: An algebraic definition of simulation between programs. In: Proc. of IJCAI'71. (1971) 481–489
13. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1995)
14. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. of ICALP'81. Volume 115 of LNCS., Springer (1981) 73–85
15. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific J. of Mathematics **5**(2) (1955) 285–309
16. Gurevich, Y., Shelah, S.: Fixed-point extensions of first order logic. Annals of Pure and Applied Logics **32** (1986) 265–280
17. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theor. Comp. Sci. **336**(1) (2005) 89–124
18. Meier, M., Schmidt, M., Lausen, G.: On chase termination beyond stratification. PVLDB **2**(1) (2009) 970–981
19. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of *DL-Lite* knowledge bases. In: Proc. of ISWC 2010. Volume 6496 of LNCS., Springer (2010) 112–128