Verification of Description Logic Knowledge and Action Bases

Babak Bagheri Hariri, Diego Calvanese, Riccardo De Masellis, Paolo Felli, Marco Montali

Abstract. We introduce description logic (DL) Knowledge and Action Bases (KAB), a mechanism that provides both a semantically rich representation of the information on the domain of interest in terms of a DL KB and a set of actions to change such information over time, possibly adding new objects. We resort to a variant of DL-Lite where UNA is not enforced and where equality between objects may be asserted and inferred. Actions are specified as sets of conditional effects, where conditions are based on epistemic queries over the KB (TBox and ABox), and effects are expressed in terms of new ABoxes. We address the verification of temporal properties expressed in a variant of first-order μ-calculus where a controlled form of quantification across states is allowed. Notably, we show decidability of verification, under a suitable restriction inspired by the notion of weak acyclicity in data exchange.

1 Introduction

Recent work in business processes, services and databases is bringing forward the need of considering both data and processes as first-class citizens in process and service design [14, 29]. In particular, the so called artifact-centric approaches, which advocate a sort of middle ground between a conceptual formalization of dynamic systems and their actual implementation, are promising to be effective in practice [12]. The verification of temporal properties in the presence of data represents a significant research challenge, since data makes the system infinite-state, and neither finite-state model checking [11] nor most of the current technique for infinite-state model checking, which mostly tackle recursion [6], apply to this case. Recently, there have been some advancements on this issue [13, 4, 5], considering a suitably constrained relational database settings.

The motivation for our work comes when we want to enrich data-intensive business processes with a semantic level. This leads us to look into how to combine first-order data, ontologies, and processes, while maintaining basic inference tasks (specifically verification) decidable. In this setting, we capture the domain of interest in terms of a semantically rich language as those provided by ontological languages based on Description Logics (DLs) [2]. Such languages natively deal with incomplete knowledge in the modeled domain. This additional flexibility comes with an added cost, however: differently from relational databases, to evaluate queries we need to resort to logical implication. Moreover incomplete information combined with the ability of evolving the system through actions results in a notoriously difficult setting [20]. In particular, due to the nature of DL assertions (which in general are not definitions but constraints on models), we get one of the most difficult kinds of domain descriptions for reasoning about actions [26], which amounts to dealing with complex forms of state constraints [18, 19]. To overcome this difficulty, virtually all effective solutions presented in the literature are based on a so-called “functional view of knowledge bases” [17]: the KB provides the ability of querying based on logical implication, and the ability of progressing it to a “new” KB through forms of updates [6, 10]. Notice that this functional view is tightly related to an epistemic interpretation of the KB [8]. Indeed our work is also related to that on Epistemic Dynamic Logic [28], and, though out of the scope of this paper, the decidability results presented here could find application in the context of that research as well.

We follow this functional view of KBs. However, a key point of our work is that at each execution step external information is incorporated into the system in form of new objects (denoted by Skolem terms), that is, our systems are not closed wrt the available information. This makes our framework particularly interesting and challenging. In particular, the presence of these objects requires a specific treatment of equality, since as the system progresses and new information is acquired, distinct object terms may be inferred to denote the same object.

Specifically, we introduce the so-called Knowledge and Action Bases (KABs). A KAB is equipped with a TBox, expressed in a variant of DL-Lite A [9, 1], which extends the core of the Web Ontology Language OWL 2 QL [9, 1] and is particularly well suited for data management. Such a TBox captures intensional information on the domain of interest, similarly to UML class diagrams or other conceptual data models, though as a software component to be used at runtime. The KAB includes also an ABox, which acts as a storage or state. The ABox maintains the data of interest, which are accessed by relying on query answering based on logical implication (certain answers). Notably, our variant of DL-Lite A is without UNA, since, as discussed above, we cannot have UNA for Skolens, and we allow for explicit equality assertions in the ABox. Technically, this breaks the first-order rewritability of DL-Lite A query answering, and requires that, in addition to the rewriting process, inference on equality is performed. As a query language, we use unions of conjunctive queries, possibly composing their certain answers through full FOL constructs. This gives rise to an epistemic query language that asks about what is “known” by the current KB [8]. The KAB then contains actions whose execution changes the state of the KB, i.e., its ABox. Such actions are specified as sets of conditional effects, where conditions are (epistemic) queries over the KB and effects are expressed in terms of new ABoxes. Actions have no static pre-conditions, instead
a process is used to specify which actions can be executed at each step. For simplicity, we model such processes as condition/action rules, where the condition is again expressed as a query over the KB.

In this setting, we address the verification of temporal/dynamic properties expressed in a first-order variant of μ-calculus \[24] [27], where atomic formulae are queries over the KB that can refer both to known and unknown objects, and where a controlled form of quantification across states is allowed. Notice that all previous decidability results on actions over DL KBs assumed that no information is coming from outside of the system, in the sense that no new objects are added while executing actions \[3] [10]. In this paper, instead, we allow for arbitrary introduction of new objects. Unsurprisingly, we show that even for very simple KABs and temporal properties, verification is undecidable. However, we also show that for a very rich class of KABs, verification is indeed decidable and reducible to finite-state model checking. To obtain this result, following \[4], we rely on recent results in data exchange on the finiteness of the chase of tuple-generating dependencies \[14].

Notice that all previous decidability results on actions over DL KBs assumed that no information is coming from outside of the system, in the sense that no new objects are added while executing actions \[3] [10]. In this paper, instead, we allow for arbitrary introduction of new objects. Unsurprisingly, we show that even for very simple KABs and temporal properties, verification is undecidable. However, we also show that for a very rich class of KABs, verification is indeed decidable and reducible to finite-state model checking. To obtain this result, following \[4], we rely on recent results in data exchange on the finiteness of the chase of tuple-generating dependencies \[14].


### 2 Knowledge Base Formalism

For expressing knowledge bases, we use DL-Lite, a variant of the DL-Lite language \[25] \[7] in which we drop the unique name assumption (UNA) \[11]. The syntax of concept and role expressions in DL-Lite is as follows:

\[
B \rightarrow N \mid \exists R \mid R \rightarrow P \mid P^-
\]

\[
C \rightarrow B \mid \neg B \quad V \rightarrow R \mid \neg R
\]

where \( N \) denotes a concept name, \( P \) a role name, and \( P^- \) an inverse role. A DL-Lite knowledge base (KB) is a pair \((T,A)\), where:

- \( T \) is a TBox, i.e., a finite set of TBox (inclusion and functionality) assertions of the form \( B \subseteq C \mid R \subseteq V \mid \{\text{funct} \, R\} \), and

- \( A \) is an ABox, i.e., a finite set of ABox (membership and equality) assertions of the form \( N(t_1) \mid P(t_1,t_2) \mid t_1 = t_2 \), where \( t_1, t_2 \) denote individuals.

As usual in DL-Lite, a TBox may contain neither (\{\text{funct} \, P\}) nor (\{\text{funct} \, P^-\}) if it contains \( R \subseteq P \) or \( R \subseteq P^- \), for some role \( R \).

We adopt the standard FOL semantics of DLs based on FOL interpretations \( \mathcal{I} = (\Delta^\mathcal{I}, \gamma^\mathcal{I}) \) such that \( c^\mathcal{I} \subseteq \Delta^\mathcal{I} \), \( N^\mathcal{I} \subseteq \Delta^\mathcal{I} \), and \( P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I} \). The semantics of the construct of, TBox and ABox assertions, and the notions of satisfaction and of model are as usual.

We also say that \( A \) is consistent wrt \( T \) if \((T,A)\) is satisfiable, i.e., admits at least one model.

Next we introduce queries. As usual (cf. OWL 2), answers to queries are formed by terms denoting individuals explicitly mentioned in the ABox. The domain of an ABox \( A \) is denoted by \( \text{ADOM}(A) \), the (finite) set of terms appearing in \( A \).

A union of conjunctive queries (UCQ) \( q \) over a KB \((T,A)\) is a FOL formula of the form \( \forall y_1,\ldots,\forall y_n.\text{conj}_1(x,y_1) \land \cdots \land \text{conj}_n(x,y_n) \), with free variables \( x \) and existentially quantified variables \( y_1,\ldots,y_n \). Each \( \text{conj}_i(x,y_i) \) in \( q \) is a conjunction of atoms of the form \( N(z), P(z,z') \), where \( N \) and \( P \) respectively denote a concept and a role name occurring in \( T \), and \( z, z' \) are constants in \( \text{ADOM}(A) \) or variables in \( x \) or \( y_i \), for some \( i \in \{1,\ldots,n\} \). The (certain) answers to \( q \) over \((T,A)\) is the set \( \text{ans}(q,T,A) \) of substitution \( q \) of the free variables of \( q \).

### 3 Knowledge and Action Bases

A Knowledge and Action Base (KAB) is a tuple \( \mathcal{K} = (T,A,\Gamma) \) where \( T \) and \( A \) form the knowledge component (or knowledge base), and \( \Gamma \) and \( \Pi \) form the action component (or action base). In practice, \( \mathcal{K} \) is a stateful device that stores the information of interest into a KB, formed by a fixed TBox \( T \) and an initial ABox \( A_0 \), which can evolve by executing actions \( \Gamma \) according to the sequencing established by process \( \Pi \). We describe such components in detail.

A TBox \( T \) is a DL-Lite TBox, used to capture the intensional knowledge about the domain of interest. Such a TBox is fixed once and for all, and does not evolve during the execution of the KAB.

An ABox \( A \) is a DL-Lite ABox, which stores the extensional information of interest. Notice that \( A_0 \) is the ABox of the initial state of the KAB, and as the KAB evolves due to the effect of actions, the ABox, which is indeed the state of the system, evolves accordingly to store up-to-date information. Through actions, we acquire new information from the external world, which results in new individuals. These individuals are denoted by (ground) Skolem terms. The presence of Skolem terms has an impact on the treatment of equality, since in principle we need to check equality w.r.t. conjunction, i.e., if \( a = b \) holds, then also \( f(a) = f(b) \) must hold. Closure w.r.t. conjunction generates an infinite number of logically implied equality assertions. However, we are going to keep such assertions implicit, computing them only when needed. Observe that, given two complex terms verifying their equality requires a PTIME computation.

### Actions

\( \Gamma \) is a finite set actions. An action \( \gamma \in \Gamma \) modifies the current ABox \( A \) by adding or deleting assertions, thus generating a new ABox \( A' \). \( \gamma \) is constituted by a signature and an effect specification.

\[q\] denotes a query occurring in \( P\) positioned in the ABox. The we also say that interpretations are dropped for brevity.

\[\text{ans}(q,T,A)\] denotes a certain answer to \( q \) over \((T,A)\).
The action signature is constituted by a name and a list of individual input parameters. Such parameters need to be instantiated with individuals for the execution of the action. Given a substitution $\theta$ for the input parameters, we denote by $\gamma\theta$ the instantiated action with the actual parameters coming from $\theta$. The effect specification consists of a set of $\{e_1, \ldots, e_n\}$ of effects, which take place simultaneously. An effect $e_i$ has the form $[q_i^+] \land Q_i^- \rightsquigarrow A_i'$, where

- $q_i^+$ is an UCQ, and $Q_i^-$ is an arbitrary ECQ whose free variables occur all among the free variables of $q_i^+$;
- $A_i'$ is a set of facts (over the alphabet of $T$) which include terms: individuals in $\text{Abox}$, free variables of $q_i^+$, and Skolem terms $f(x)$ having as arguments free variables $x$ of $q_i^+$.

Given the current ABox $A$ of $K$ and a substitution $\theta$ for the parameters of the action $\gamma$, the new state $A'$ resulting from firing the action $\gamma \theta$ on the state $A$ is computed as follows: (i) each effect $e_i \in \gamma \theta$ extracts from $A$ the set $\text{ANS}(\{q_i^+\} \land Q_i^- \theta, T, A)$ of tuples of terms in $\text{ADOM}(A)$, and for each such tuple $\sigma$ asserts a set $A_i\theta\sigma$ of facts obtained from $A_i\theta$ by applying the substitution $\sigma$ for the free variables of $q_i^+$. For each Skolem term $f(x)\theta$ appearing in $A_i\theta'$, a new ground term is introduced having the form $f(x)\theta x$. These terms represent new “constants” denoting “unknown” individuals. We denote by $e_i(\theta)(A)$ the overall set of facts, i.e., $e_i(\theta)(A) = \bigcup_{\sigma \in \text{ANS}(\{q_i^+\} \land Q_i^- \theta, T, A)} A_i\theta\sigma$. (ii) Moreover, let $\text{EQ}(A) = \{t_1 = t_2 \mid \{t_1,t_2\} \in \text{ANS}(x=[x_1], T, A)\}$. Observe that, due to the semantics of queries, the terms in $\text{EQ}(A)$ must appear explicitly in $\text{ADOM}(A)$, that is, the possibly infinite number of equalities due to congruence do not appear in $\text{EQ}(A)$, though they are logically implied. The overall effect of the action $\gamma \theta$ parameter substitution $\theta$ over $A$ is the new ABox $A' = \text{DO}(T, A, \gamma \theta)$ where $\text{DO}(T, A, \gamma \theta) = e_i(\theta)(A) \cup \bigcup_{\sigma \in \text{EQ}(A)} e_i(\theta)(A)$.

Let us make some observations on such effects. The effects of an action are a form of update of the previous state $A$ and not of belief revision $\text{REF}(A)$. 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. Skolem terms introduced by actions effects can be thought of witnesses of new information coming from an external user/environment when executing the action. Their presence makes the domain of the ABoxes obtained by executing actions continuously changing. For simplicity, we do not make any persistence (or frame) assumption in our formalization (except for equality) $\text{REF}(A)$.

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 $P$. We do have a persistence assumption on equalities, we implicitly copy all equalities holding in the current state to the new one. This implies that, as the system evolves, we acquire new information on equalities between terms, but never lose equality information already acquired.

Process. The process component of a KAB is a possibly nondeterministic program that uses the KAB ABoxes to store its (intermediate and final) computation results, and the actions in $T$ as atomic instructions. The ABoxes can be arbitrarily queried through the KAB TBox.

---

5 We disregard a specific treatment of the output to the user, and assume instead that she can freely pose queries over the KB, extracting implicit or explicit information from the states through which the KAB evolves.

6 The UCQ-ECQ division is a convenience to have readily available the positive part of the condition.

7 Our approach sidesteps the semantical and computational difficulties of description logic knowledge base update $\text{REF}(A)$. Adopting such forms of update in our setting is an interesting research venue.
unmasked, i.e., it is known that $s$ is equal to his alter ego.

Challenge($v, s$):
\[
\{ \text{Villain}(v) \land \text{Superhero}(s) \land \exists p. \text{alterEgo}(s, p) \land \text{livesIn}(p, sc) \} \Rightarrow \{ \text{livesIn}(v, sc), \text{enemy}(v, s), \; \text{CopyAll} \}
\]
states that when villain $v$ challenges superhero $s$ and has not defeated him, next he lives in the same city as $s$ and is enemy of $s$.

ThreatenCity($v, c$):
\[
\{ \text{Villain}(v) \land \text{Superhero}(s) \land \exists p. \text{alterEgo}(s, p) \land \text{livesIn}(p, c) \} \Rightarrow \{ \text{enemy}(v, s), \; \text{CopyAllExceptEnemy} \}
\]
states that when villain $v$ threatens city $c$, then he becomes an enemy of all and only superheros that live in $c$.

Consider an initial ABox $A_0 = \{ \text{Superhero}(\text{batman}), \text{Villain}(\text{joker}), \text{alterEgo}(\text{batman}, \text{bruce}), \text{livesIn}(\text{bruce}, \text{gotham}) \}$, livesIn($\text{gotham}, \text{gotham}$), livesIn($\text{joker}, \text{city1}$)). In this state, bruce and batman live in the same city, and batman is the alterego of bruce but it is not known whether they denote the same individual. Executing Challenge($\text{jetman}, \text{batman}$) in $A_0$ generates a new ABox with added facts $\text{enemy}(\text{jetman}, \text{batman})$, livesIn($\text{joker}, \text{gotham}$), and gotham $\Rightarrow$ city1 is implied by the functionality on livesIn.

A process $\Pi$ might include the following rules:
\[
\begin{align*}
\text{Character}(p) & \Rightarrow \text{BecomeSH}(p, c), \\
\text{Superhero}(s) \land \text{Character}(c) & \Rightarrow \text{Unmask}(s, c), \\
\text{enemy}(v, s) & \Rightarrow \text{Fight}(v, s), \\
\text{Villain}(v) \land \text{Superhero}(s) & \Rightarrow \text{Challenge}(v, s), \\
\text{Villain}(v) \land \text{City}(c) & \Rightarrow \text{livesIn}(v', c) \land -\exists v'((\text{Villain}(v') \land \text{livesIn}(v', c)) \land -[v = v']) \Rightarrow \text{ThreatenCity}(v, c)
\end{align*}
\]
For instance, the first one states that a character can become a superhero if it is not already one.

5 Verification

To specify dynamic properties over a semantic artifacts, we use a first-order variant of $\mu$-calculus [27, 24]. (Temporal) $\mu$-calculus is virtually the most powerful temporal logic used for model checking of finite-state transition systems, and is able to express both linear time logics such as LTL and PSL, and branching time logics such as CTL and CTL*. The main characteristic of $\mu$-calculus is its ability of expressing directly least and greatest fixpoints of (predicate transformer) operators formed using formulae relating the current state to the next one. By using such fixpoint constructs one can easily express sophisticated properties defined by induction or co-induction. This is the reason why virtually all logics used in verification can be considered as fragments of $\mu$-calculus. Technically, $\mu$-calculus separates local properties, asserted on the current state or on states that are immediate successors of the current one, and properties talking about states that are arbitrarily far away from the current one [27]. The latter are expressed through the use of fixpoints.

In this work, we use a first-order variant of $\mu$-calculus, where we allow local properties to be expressed as EQL queries, and at the same time we allow arbitrary first-order quantification across states. Given the nature of EQL queries used for formulating local properties, first-order quantification must range over terms denoting individuals. Formally, we introduce the logic $\mu L_A$ defined as follows:
\[
\Phi \longrightarrow Q \mid -\Phi \mid \Phi_1 \land \Phi_2 \mid \exists x. \Phi \mid (\neg \Phi) \land Z \mid \mu Z. \Phi
\]
where $Q$ is a possibly open EQL query, and $Z$ is a second order predicate variable (of arity 0). We make use of the following abbreviations: $\forall x. \Phi = \neg(\exists x. \neg \Phi)$, $\Phi_1 \lor \Phi_2 = \neg(\neg \Phi_1 \land \neg \Phi_2)$, $[-\Phi] = \neg\neg(\neg \Phi)$, and $\nu Z. \Phi = \neg\mu Z. \neg\Phi[Z/\neg Z]$. The formulae $\mu Z. \Phi$ and $\nu Z. \Phi$ respectively denote the least and greatest fixpoint of the formula $\Phi$ (seen as the predicate transformer $AZ. \Phi$). As usual in $\mu$-calculus, formulae of the form $\mu Z. \Phi$ (and $\nu Z. \Phi$) must obey to the syntactic monotonicity of $\Phi$ wrt $Z$, which states that every occurrence of the variable $Z$ in $\Phi$ must be within the scope of an even number of negation symbols. This ensures that the least fixpoint $\mu Z. \Phi$ (as well as the greatest fixpoint $\nu Z. \Phi$) always exists.

The semantics of $\mu L_A$ formulae is defined over possibly infinite transition systems $\langle U, T, \Sigma, s_0, \text{abox}, \Rightarrow \rangle$. Since $\mu L_A$ also contains formulae with both individual and predicate free variables, given a transition system $\Sigma$, we introduce an individual variable valuation $v$, i.e., a mapping from individual variables $x$ to $U$, and a predicate variable valuation $V$, i.e., a mapping from the predicate variables $Z$ to a subset of $\Sigma$. With these three notions in place, we assign meaning to formulae by associating to $\Sigma, v, V$ and $\nu z. \Phi$ an extension function ($\nu_z \Phi$), which maps formulae to subsets of $\Sigma$. Formally, the extension function ($\nu_z \Phi$) is defined inductively as follows:
\[
\begin{align*}
(Q^\Sigma)^v_V &= \{ s \in \Sigma \mid \text{ans}(Qv, T, \text{abox}(s)) = \text{true} \}, \\
(\Phi_1 \land \Phi_2)^\Sigma_v_V &= (\Phi_1)^\Sigma_v_V \land (\Phi_2)^\Sigma_v_V, \\
(\exists x. \Phi)^\Sigma_v_V &= \{ s \in \Sigma \mid \exists t. s \in \text{abox}(s) \text{ and } s \in \Phi^\Sigma_v_V \}, \\
(-\Phi)^\Sigma_v_V &= \Sigma \setminus (\Phi)^\Sigma_v_V, \\
(\Phi_1 \lor \Phi_2)^\Sigma_v_V &= (\Phi_1)^\Sigma_v_V \lor (\Phi_2)^\Sigma_v_V, \\
(V)^\Sigma_v_V &= V(Z), \\
(\mu Z. \Phi)^\Sigma_v_V &= \{ s \in \Sigma \mid Z/s \Rightarrow s' \text{ and } s' \notin (\Phi)^\Sigma_v_V \}, \\
(\nu Z. \Phi)^\Sigma_v_V &= \bigcap \{ E \subseteq \Sigma \mid \Phi^\Sigma_v_V[E] \subseteq E \}
\end{align*}
\]
Here $Qv$ stands for the query obtained from $Q$ by substituting its free variables according to $\nu$. Intuitively, $(\cdot)^{1}_{\nu,v}$ assigns to such constructs the following meanings:

- The boolean connectives have the expected meaning.
- The quantification of individuals is done over the objects of the “current” ABox.
- The extension of $\langle \cdot \rangle \Phi$ consists of the states $s$ such that for some state $s'$ with $s \Rightarrow s'$, we have that $\Phi$ holds in $s'$, while the extension of $\neg \Phi$ consists of the states $s$ such that for all states $s'$ with $s \Rightarrow s'$, $\Phi$ holds in $s'$.
- The extension of $\mu \Phi$ is the smallest subset $\subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subsection{Theorem 3} For two states $s_1, s_2$, if $s_1 \in (\Phi)^{1}_{\nu,v}$ then $s_2 \in (\Phi)^{2}_{\nu,v}$. Proof. The proof is analogous to the standard proof of bisimulation invariance of $\mu$-calculus \cite{21}, though taking into account our bisimulation, which guarantees that ECQs are evaluated identically over bisimilar states.

We observe that in this invariance result, we make no use of the restriction on existentially quantified individuals belonging to the “current” active domain. However, we enforce this restriction to be able to talk only about individuals that are explicitly mentioned in the ABoxes of the transition system (i.e., in the active domain of the transition system) and not those that are implicitly present because of the congruence of equalities.

Making use of such a notion of bisimulation, we can redefine the transition system generated by KAB $K = (T, A_0, \Gamma, \Pi)$ while maintaining bisimilarity, by modifying the definition of $\mathcal{K}$ as follows. (i) We modify do$(t)$ so that no Skolem term $t'$ is introduced in the generated ABox $A'$ if in the current ABox $A$ there already is a term $t$ such that $(T, A) \models t = t'$. (ii) If the ABox $A' = \text{do}(T, axob(s), \gamma \theta)$ obtained from the current state $s$ is logically equivalent to the ABox $axob(s''')$, for some already generate state $s''$, we do not generate a new state, but simply add $s \Rightarrow s''$ to $\mathcal{K}$.

6 Weakly Acyclic KABs

Verification of KABs is undecidable in general. Indeed, we have:

\textbf{Theorem 4} Verification of CTL reachability-like formulas of the form $\mu Z. (N(a) \lor \langle \cdot \rangle Z)$ (with $N$ an atomic concept and $a$ an individual occurring in $A_0$) on KABs with empty TBoxes and actions that make use only of UCQs (no negation nor equality) is undecidable.

Proof. By reduction to answering boolean UCQs in a relational database under a set of tuple-generating dependencies (TDGs), which is undecidable \cite{13}.

Hence it is of interest to isolate interesting special cases in which verification is decidable. Next, we introduce a notable class of KABs that enjoys such a property. In particular, we show that a suitable syntactic restriction, which resembles the notion of weak acyclicity in data exchange \cite{15},\footnote{Note that all terms that are present in the current ABox are preserved in the new ABox, together with equalities between terms.} guarantees boundedness of ABoxes generated by the execution of the nKAB, and in turn decidability of verification.

To do so, we introduce the edge-labeled directed dependency graph of a KAB $K = (T, A_0, \Gamma, \Pi)$, defined as follows. Nodes, called positions, are obtained from the TBox $T$: there is a node for every concept name $N$ in $T$, and two nodes for every role name $P$ in $T$, corresponding to the domain and to the range of $P$. Edges are drawn by considering every effect specification $[q^{+}] \land Q^{-} \leadsto A'$ of each action contained in $\Gamma$, tracing how values are copied or contribute to generate new values as the system progresses. In particular, let $p$ be a position corresponding to a concept/role component in the rewriting $\text{rew}(q^{-})$ of $q^{-}$ with variable $x$. For every position $p'$ in $A'$ with the same variable $x$, we include a normal edge $p \rightarrow p'$. For every position $p''$ in $A'$ with a Skolem term $f(t)$ such that $x \in t$, we include a special edge $p \rightarrow p''$. We say that $K$ is weakly-acyclic if its dependency graph has no cycle going through a special edge.

\footnote{We use the original definition of weak acyclicity. However, our results depend only on the ability of finding a finite bound for the chase. So, other variants of weak acyclicity, such as \cite{21},\cite{22}, can also be adopted.}
The restriction imposed by weak acyclicity (or variants) is not too severe, and in many real cases KABs are indeed weakly acyclic or can be transformed into weakly acyclic ones at cost of redesign. Indeed, if a KAB is not weakly acyclic, it indefinitely generates new values from the old ones, which then depend on a chain of unboundedly many previous values. In other words, current values depend on old ones, which then depend on a chain of unbounded values. In this case, the KAB can in principle be rewritten into a weakly acyclic one. While such unbounded systems exist in theory, they are not typically required in practice. How to systematically transform systems into weakly acyclic ones remains an open issue.

7 Conclusions

In this paper we have studied verification of knowledge and action bases, which are dynamic systems constituted by a knowledge base expressed in description logic, and by an action specification that changes the knowledge base over time. We have obtained an interesting decidability result by relying on the notion of weak acyclicity, based on a connection with the theory of chase of TGDs in relational databases. With this at hand, it becomes of interest to study refined action specifications that guarantee better elaboration tolerance (addressing the frame, ramification, and qualification problems) [20].

Acknowledgements. This research has been partially supported by the ICT Collaborative Project ACSI (Artifact-Centric Service Interoperation), funded by the EU under FP7 ICT Call 5, 2009.1.2, grant agreement No. FP7-257593.

REFERENCES


doi:10.3233/978-1-61499-098-7-103

The final publication is available at ebooks.iospress.nl