Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy
Tel: +39 04710 16000, fax: +39 04710 16009

KRDB Research Centre Technical Report:

Verification of Inconsistency-Aware Knowledge and Action Bases (Extended Version)

Diego Calvanese[1], Evgeny Kharlamov[1], Marco Montali[1], Ario Santoso[1], Dmitriy Zheleznyakov[1]

Affiliation	1: KRDB Research Centre for Knowledge and Data Free University of Bozen-Bolzano *lastname*@inf.unibz.it
Corresponding author	Marco Montali: montali@inf.unibz.it
Number	KRDB13-2
Date	April 23, 2013
URL	http://www.inf.unibz.it/krdb/

Abstract

Description Logic Knowledge and Action Bases (KABs) have been recently introduced as a mechanism that provides 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 introducing new objects. In this setting, decidability of verification of sophisticated temporal properties over KABs, expressed in a variant of first-order μ-calculus, has been shown. However, the established framework treats inconsistency in a simplistic way, by rejecting inconsistent states produced through action execution. We address this problem by showing how inconsistency handling based on the notion of repairs can be integrated into KABs, resorting to inconsistency-tolerant semantics. In this setting, we establish decidability and complexity of verification. We then show how in this approach the temporal properties can be enriched with meta-level requests about the sources of inconsistency, and we extend decidability and complexity of verification accordingly.

Acknowledgments

The authors are supported by the EU project ACSI (FP7-ICT-257593) and Optique (FP7-IP-318338). Kharlamov was also supported by the ERC grant Webdam, agreement n. 226513.

Description Logic Knowledge and Action Bases (KABs) have been recently introduced as a mechanism that provides 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 introducing new objects. In this setting, decidability of verification of sophisticated temporal properties over KABs, expressed in a variant of first-order μ-calculus, has been shown. However, the established framework treats inconsistency in a simplistic way, by rejecting inconsistent states produced through action execution. We address this problem by showing how inconsistency handling based on the notion of repairs can be integrated into KABs, resorting to inconsistency-tolerant semantics. In this setting, we establish decidability and complexity of verification.

1. Introduction

Recent work in knowledge representation and databases has addressed the problem of dealing with the combination of knowledge, processes and data in the design of complex enterprise systems [11, 23, 1, 8, 19]. The verification of temporal properties in this setting represents a significant research challenge, since data and knowledge makes the system infinite-state, and neither finite-state model checking [10] nor most of the current techniques for infinite-state model checking [4] apply to this case.

Along this line, *Knowledge and Action Bases* (*KABs*) [1] have have been recently introduced as a mechanism that provides a semantically rich representation of the information on the domain of interest in terms of a Description Logic (DL) KB and a set of actions to change such information over time, possibly introducing new objects. In this setting, decidability of verification of sophisticated temporal properties over KABs, expressed in a variant of first-order μ-calculus, has been shown.

However, KABs and the majority of approaches dealing with verification in this complex setting assume a rather simple treatment of inconsistency resulting as an effect of action execution: inconsistent states are simply rejected (see, e.g., [12, 11, 2]). In general, this is not satisfactory, since the inconsistency may affect just a small portion of the entire KB, and should be treated in a more careful way. Starting from this observation, in this work we leverage on the research on instance-level evolution of knowledge bases [24, 13, 16, 9], and, in particular, on the notion of knowledge base repair [18], in order to make KABs inconsistency-aware. In particular, we present a novel setting that extends KABs by assuming the availability of a repair service that is able to compute, from an inconsistent knowledge base resulting from the execution of an action, one or more *repairs*, in which the inconsistency has been removed with a "minimal" modification to the existing knowledge. This allows us to incorporate, in the temporal verification formalism, the possibility of quantifying over repairs. Notably, our novel setting is able to deal with both deterministic semantics for repair, in which a single repair is computed from an inconsistent knowledge base, and non-deterministic ones, by simultaneously taking into account all possible repairs. We show how the techniques developed for KABs extend to this inconsistency-aware setting, preserving both decidability and complexity results, under the same assumptions required in KABs for decidability.

We also show how our setting is able to accommodate meta-level information about the sources of inconsistency at the intentional level, so as to allow them to be queried when verifying temporal properties of the system. The decidability and complexity results for verification carry over to this extended setting as well.

The proofs of all presented theorems are contained in the appendix.

2. Preliminaries

2.1. DL-Lite$_\mathcal{A}$ Knowledge Bases

For expressing knowledge bases, we use *DL-Lite*$_\mathcal{A}$ [21, 5]. The syntax of *concept* and *role expressions* in *DL-Lite*$_\mathcal{A}$ is as follows

$$B \longrightarrow N \mid \exists R \qquad\qquad R \longrightarrow P \mid P^-$$

where N denotes a *concept name*, P a *role name*, and P^- an *inverse role*. A *DL-Lite*$_\mathcal{A}$ *knowledge base* (KB) is a pair (T, A), where: *(i)* A is an Abox, i.e., a finite set of *ABox membership assertions* of the form $N(t_1) \mid P(t_1, t_2)$, where t_1, t_2 denote individuals *(ii)* T is a TBox, i.e., $T = T_p \uplus T_n \uplus T_f$, with T_p a finite set of *positive inclusion assertions* of the form $B_1 \sqsubseteq B_2$, T_n a finite set of *negative inclusion assertions* of the form $B_1 \sqsubseteq \neg B_2$, and T_f a finite set of *functionality assertions* of the form (funct R).

We adopt the standard FOL semantics of DLs based on FOL interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ such that $c^\mathcal{I} \in \Delta^\mathcal{I}$, $N^\mathcal{I} \subseteq \Delta^\mathcal{I}$, 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 T*-consistent* if (T, A) is satisfiable, i.e., admits at least one model, otherwise we say A is T*-inconsistent*.

Queries. As usual (cf. OWL 2 QL), answers to queries are formed by terms denoting individuals explicitly mentioned in the ABox. The *domain of an ABox* A, denoted by $\text{ADOM}(A)$, is 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 $\bigvee_{1 \le i \le n} \exists \vec{y_i}.conj_i(\vec{x}, \vec{y_i})$ with free variables $\vec{x}$ and existentially quantified variables $\vec{y_1}, \ldots, \vec{y_n}$. Each $conj_i(\vec{x}, \vec{y_i})$ 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 $\vec{x}$ or $\vec{y_i}$, for some $i \in \{1, \ldots, n\}$. The *(certain) answers* to q over (T, A) is the set $ans(q, T, A)$ of substitutions σ of the free variables of q with constants in $\text{ADOM}(A)$ such that $q\sigma$ evaluates to true in every model of (T, A). If q has no free variables, then it is called *boolean* and its certain answers are either true or false.

We compose UCQs using ECQs, i.e., queries of the query language *EQL-Lite*(UCQ) [6], which is the FOL query language whose atoms are UCQs evaluated according to the certain answer semantics above. An *ECQ* over T and A is a possibly open formula of the form

$$Q \;:=\; [q] \mid \neg Q \mid Q_1 \wedge Q_2 \mid \exists x.Q$$

where q is a UCQ. The *answer to* Q *over* (T, A), is the set $\text{ANS}(Q, T, A)$ of tuples of constants in $\text{ADOM}(A)$ defined by composing the certain answers $ans(q, T, A)$ of UCQs q through first-order constructs, and interpreting existential variables as ranging over $\text{ADOM}(A)$.

Finally, we recall that *DL-Lite*$_\mathcal{A}$ enjoys the *FO rewritability* property, which states that for every UCQ q, $ans(q, T, A) = ans(rew(q), \emptyset, A)$, where $rew(q)$ is a UCQ computed by the reformulation algorithm in [5]. Notice that this algorithm can be extended to ECQs [6], and that its effect is to "compile away" the TBox.

2.2. Knowledge and Action Bases

We recall the notion of *Knowledge and Action Bases (KABs)*, as introduced in [1]. In the following, we make use of a countably infinite set $\mathcal{C}$ of constant to denote all possible value in the system. Moreover, we also make use of a finite set $\mathcal{F}$ of functions that represent service calls, and can be used to inject fresh values into the system.

A KAB is a tuple $\mathcal{K} = (T, A_0, \Gamma, \Pi)$ where T and A_0 form the knowledge base (KB), and Γ and Π form the action base. Intuitively, the KB maintains the information of interest. It is formed by a fixed *DL-Lite*$_\mathcal{A}$ TBox T and an initial *T-consistent DL-Lite*$_\mathcal{A}$ ABox A_0. A_0 represents the initial state of the system and, differently from T, it evolves and incorporates new information from the external world by executing actions Γ, according to the sequencing established by process Π. Γ 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'. γ 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 instantiated with individuals for the execution of the action. Given a substitution θ for the input parameters, we denote by $\gamma\theta$ the instantiated action with the *actual* parameters coming from θ. The *effect specification* consists of a set $\{e_1, \ldots, e_n\}$ of effects, which take place simultaneously. An *effect* e_i has the form $[q_i^+] \wedge Q_i^- \rightsquigarrow A_i'$, where: *(i)* q_i^+ is an UCQ, and Q_i^- is an arbitrary ECQ whose free variables occur all among the free variables of q_i^+; *(ii)* A_i' is a set of facts (over the alphabet of T) which include as terms: individuals in A_0, free variables of q_i^+, and Skolem terms $f(\vec{x})$ having as arguments free variables $\vec{x}$ of q_i^+. The distinction between q_i^+ and Q_i^- is needed for technical reasons (see Appendix E).

The process Π is a finite set of condition/action rules. A *condition/action rule* $\pi \in \Pi$ is an expression of the form $Q \mapsto \gamma$, where γ is an action in Γ and Q is an ECQ over T, whose free variables are exactly the parameters of γ. The rule expresses that, for each tuple σ for which condition Q holds, the action γ with actual parameters σ *can* be executed.

Example 2.2.1. *$\mathcal{K} = (T, A_0, \Gamma, \Pi)$ is a KAB defined as follows: (i) $T = \{C \sqsubseteq \neg D\}$, (ii) $A_0 = \{C(a)\}$, (iii) $\Gamma = \{\gamma_1, \gamma_2\}$ with $\gamma_1() : \{C(x) \rightsquigarrow D(x), C(x)\}$ and $\gamma_2(p) : \{C(p) \rightsquigarrow G(f(p))\}$, (iv) $\Pi = \{true \mapsto \gamma_1, C(y) \mapsto \gamma_2(y)\}$.* □

3. Verification of Standard KABs

We are interested in verifying temporal/dynamic properties over KABs. To this aim, we fix a countably infinite set $\mathcal{C}$ of individual constants (also called values), which act as standard names, and finite set of distinguished constants $\mathcal{C}_0 \subset \mathcal{C}$. Then, we define the execution semantics of a KAB in terms of a possibly infinite-state *transition system*. More specifically, we consider transition systems of the form $(\mathcal{C}, T, \Sigma, s_0, abox, \Rightarrow)$, where: *(i)* T is a TBox; *(ii)* Σ is a set of states; *(iii)* $s_0 \in \Sigma$ is the initial state; *(iv)* *abox* is a function that, given a state $s \in \Sigma$, returns an ABox associated to s, which has as individuals values of $\mathcal{C}$ and conforms to T; *(v)* $\Rightarrow \subseteq \Sigma \times \Sigma$ is a transition relation between pairs of states.

The standard execution semantics for a KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$ is obtained starting from A_0 by nondeterministically applying every executable action with corresponding legal parameters, and considering each possible value returned by applying the involved service calls. Notice that this is radically different from [1], where service calls are not evaluated when constructing the transition system. The executability of an action with fixed parameters does not only depend on the process Π, but also on the T-consistency of the ABox produced by the application of the action: if the resulting ABox is T-inconsistent, the action is considered as non executable with the chosen parameters.

We consider *deterministic* services, i.e., services that return always the same value when called with the same input parameters. Nondeterministic services can be seamlessly added without affecting our technical results. To ensure that services behave deterministically, we recast the approach in [2] to the semantic setting of KABs, keeping track, in the states of the transition system generated by $\mathcal{K}$, of all the service call results accumulated so far. To do so, we introduce the set of (Skolem terms representing) service calls as $\mathbb{SC} = \{f(v_1, \ldots, v_n) \mid f/n \in \mathcal{F} \text{ and } \{v_1, \ldots, v_n\} \subseteq \mathcal{C}\}$, and define a *service call map* as a partial function $m : \mathbb{SC} \to \mathcal{C}$.

A *state* of the transition system generated by $\mathcal{K}$ is a pair $\langle A, m \rangle$, where A is an ABox and m is a service call map. Let $\gamma(p_1, \ldots, p_r) : \{e_1, \ldots, e_k\}$ be an action in Γ with parameters $p_1, \ldots, p_r$, and $e_i = [q_i^+] \wedge Q_i^- \rightsquigarrow E_i$. Let σ be a substitution for $p_1, \ldots, p_r$ with values taken from $\mathcal{C}$. We say that σ is *legal* for γ in state $\langle A, m \rangle$ if there exists a condition-action rule $Q \mapsto \gamma$ in Π such that $\langle p_1, \ldots, p_r \rangle \sigma \in \textsc{ans}(Q, A)$. We denote with $\textsc{do}(T, A, \gamma\sigma)$ the set of facts obtained by evaluating the effects of action γ with parameters σ on ABox A, so as to *progress* (cf. planning [17]) the system from the current state to the next:

$$\textsc{do}(T, A, \gamma\sigma) = \bigcup_{[q_i^+] \wedge Q_i^- \rightsquigarrow E_i \text{ in } \gamma} \; \bigcup_{\rho \in \textsc{ans}(([q_i^+] \wedge Q_i^-)\sigma, T, A)} E_i \sigma \rho$$

The returned set is the union of the results of applying the effects specifications in

γ, where the result of each effect specification $[q_i^+] \wedge Q_i^- \rightsquigarrow E_i$ is, in turn, the set of facts $E_i\sigma\rho$ obtained from $E_i\sigma$ grounded on all the assignments ρ that satisfy the query $[q_i^+] \wedge Q_i^-$ over A.

Note that $\textsc{do}()$ generates facts that use values from the domain $\mathcal{C}$, but also Skolem terms, which model service calls. For any such set of facts E, we denote with $\textsc{calls}(E)$ the set of calls it contains, and with $\textsc{evals}(T, A, \gamma\sigma)$ the set of substitutions that replace all service calls in $\textsc{do}(T, A, \gamma\sigma)$ with values in $\mathcal{C}$:

$$\textsc{evals}(T, A, \gamma\sigma) = \{\theta \mid \theta \text{ is a total function } \theta : \textsc{calls}(\textsc{do}(T, A, \gamma\sigma)) \to \mathcal{C}\}.$$

Each substitution in $\textsc{evals}(T, A, \gamma\sigma)$ models the simultaneous evaluation of all service calls, returning results arbitrarily chosen from $\mathcal{C}$.

Example 3.0.2. *Consider our running example (Example 2.2.1). Starting from A_0, the execution of γ_1 would produce $A' = \{D(a), C(a)\}$, which is T-inconsistent. Thus, the execution of γ_1 in A_0 should either be rejected or its effect should be repaired (cf. Section 4). The execution of γ_2 with legal parameter a instead produces $A'' = \{G(c)\}$ when the service call $f(a)$ returns c. A'' is T-consistent, and $\gamma_2(a)$ is therefore executable in A_0.* □

Given a KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$, we employ $\textsc{do}()$ and $\textsc{evals}()$ to define a transition relation $\textsc{exec}_\mathcal{K}$ connecting two states through the application of an action with parameter assignment. In particular, given an action with parameter assignment $\gamma\sigma$, we have $\langle\langle A, m\rangle, \gamma\sigma, \langle A', m'\rangle\rangle \in \textsc{exec}_\mathcal{K}$ if the following holds: *(i)* σ is a legal parameter assignment for γ in state $\langle A, m\rangle$, according to Π; *(ii)* there exists $\theta \in \textsc{evals}(T, A, \gamma\sigma)$ such that θ and m agree on the common values in their domains (so as to realize the deterministic service semantics); *(iii)* $A' = \textsc{do}(T, A, \gamma\sigma)\theta$; *(iv)* $m' = m \cup \theta$ (i.e., the history of issued service calls is updated).

Standard transition system. The *standard transition system* $\Upsilon^{\mathrm{S}}_\mathcal{K}$ for KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$ is a (possibly infinite-state) transition system $(\mathcal{C}, T, \Sigma, s_0, abox, \Rightarrow)$ where: *(i)* $s_0 = \langle A_0, \emptyset\rangle$; *(ii)* $abox(\langle A, m\rangle) = A$; *(iii)* Σ and $\Rightarrow$ are defined by simultaneous induction as the smallest sets satisfying the following properties: *(i)* $s_0 \in \Sigma$; *(ii)* if $\langle A, m\rangle \in \Sigma$, then for all actions γ in Γ, for all substitutions σ for the parameters of γ and for all $\langle A', m'\rangle$ such that A' is *T-consistent* and $\langle\langle A, m\rangle, \gamma\sigma, \langle A', m'\rangle\rangle \in \textsc{exec}_\mathcal{K}$, we have $\langle A', m'\rangle \in \Sigma$ and $\langle A, m\rangle \Rightarrow \langle A', m'\rangle$. We call S-KAB a KAB interpreted under the standard execution semantics.

Example 3.0.3. *Consider $\mathcal{K}$ of Example 2.2.1 and its standard transition system $\Upsilon^{\mathrm{S}}_\mathcal{K}$. As discussed in Example 3.0.2, in state $s_0 = \langle A_0, \emptyset\rangle$ only γ_2 is applicable with parameter a. Since $\textsc{do}(T, A_0, \gamma_2(a)) = \{G(f(a))\}$, $\Upsilon^{\mathrm{S}}_\mathcal{K}$ contains infinitely many successors for s_0, each of the form $\langle\{G(x)\}, \{f(a) \mapsto x\}\rangle$, where x is arbitrarily substituted with a specific value picked from $\mathcal{C}$.* □

Verification Formalism. To specify sophisticated temporal properties over KABs, we resort to the first-order variant of μ-calculus [22, 20] defined in [1]. This variant, here called $\mu\mathcal{L}_A^{\mathrm{EQL}}$, exploits EQL to query the states, and supports a particular form of first-order quantification across states: quantification ranges over the individuals

explicitly present in the current active domain, and can be arbitrarily referred to in later states of the systems. Formally, $\mu\mathcal{L}_A^{\text{EQL}}$ is defined as follows:

$$\Phi \;:=\; Q \;\mid\; \neg\Phi \;\mid\; \Phi_1 \wedge \Phi_2 \;\mid\; \exists x.\Phi \;\mid\; \langle\!\leftarrow\!\rangle\Phi \;\mid\; Z \;\mid\; \mu Z.\Phi$$

where Q is a possibly open EQL query that can make use of the distinguished constants in $\mathcal{C}_0$, and Z is a second order predicate variable (of arity 0). We make use of the following abbreviations: $\forall x.\Phi = \neg(\exists x.\neg\Phi)$, $\Phi_1 \vee \Phi_2 = \neg(\neg\Phi_1 \wedge \neg\Phi_2)$, $[\!\leftarrow\!]\Phi = \neg\langle\!\leftarrow\!\rangle\neg\Phi$, and $\nu Z.\Phi = \neg\mu Z.\neg\Phi[Z/\neg Z]$.

The semantics of $\mu\mathcal{L}_A^{\text{EQL}}$ formulae is defined over transition systems $\langle\mathcal{C}, T, \Sigma, s_0, abox, \Rightarrow\rangle$. Since $\mu\mathcal{L}_A^{\text{EQL}}$ contains formulae with both individual and predicate free variables, given a transition system Υ, we introduce an individual variable valuation v, i.e., a mapping from individual variables x to $\mathcal{C}$, and a predicate variable valuation V, i.e., a mapping from the predicate variables Z to a subset of Σ. All the language primitives follow the standard μ-calculus semantics, apart from the two listed below [1]:

$$\begin{aligned}(Q)^{\Upsilon}_{v,V} &= \{s \in \Sigma \mid \textsc{ans}(Qv, T, abox(s)) = true\}\\ (\exists x.\Phi)^{\Upsilon}_{v,V} &= \{s \in \Sigma \mid \exists d.d \in \textsc{adom}(abox(s))\\ &\qquad\qquad \text{and } s \in (\Phi)^{\Upsilon}_{v[x/d],V}\}\end{aligned}$$

Here, Qv stands for the query obtained from Q by substituting its free variables according to v. When Φ is a closed formula, $(\Phi)^{\Upsilon}_{v,V}$ does not depend on v or V, and we denote the extension of Φ simply by $(\Phi)^{\Upsilon}$. A closed formula Φ holds in a state $s \in \Sigma$ if $s \in (\Phi)^{\Upsilon}$. We call *model checking* verifying whether $s_0 \in (\Phi)^{\Upsilon}$, and we write in this case $\Upsilon \models \Phi$.

Decidability of verification. We are interested in studying the verification of $\mu\mathcal{L}_A^{\text{EQL}}$ properties over S-KABs. We can easily recast the undecidability result in [1] to the case of S-KABs, obtaining that verification is undecidable even for the very simple temporal reachability property $\mu Z.(\mathsf{N}(a) \vee \langle\!\leftarrow\!\rangle Z)$, with N atomic concept and $a \in \mathcal{C}$.

Despite this undecidability result, we can isolate an interesting class of KABs that enjoys verifiability of arbitrary $\mu\mathcal{L}_A^{\text{EQL}}$ properties through finite-state abstraction. This class is based on a semantic restriction named *run-boundedness* [2]. Given an S-KAB $\mathcal{K}$, a run $\tau = s_0 s_1 \cdots$ of $\Upsilon_{\mathcal{K}}^{\text{S}}$ is *bounded* if there exists a finite bound b s.t. $|\bigcup_{s \text{ state of } \tau} \textsc{adom}(abox(s))| < b$. We say that $\mathcal{K}$ is *run-bounded* if there exists a bound b s.t. every run τ in $\Upsilon_{\mathcal{K}}^{\text{S}}$ is bounded by b.

Theorem 3.0.4. *Verification of $\mu\mathcal{L}_A^{\text{EQL}}$ properties over run-bounded S-KABs is decidable, and can be reduced to finite-state model checking of propositional μ-calculus.*

The crux of the proof is to show, given a run-bounded S-KAB $\mathcal{K}$, how to construct an *abstract transition system* $\Theta_{\mathcal{K}}^{\text{S}}$ that satisfies exactly the same $\mu\mathcal{L}_A^{\text{EQL}}$ properties as the original transition system $\Upsilon_{\mathcal{K}}^{\text{S}}$. This is done by introducing a suitable bisimulation relation, and defining a construction of $\Theta_{\mathcal{K}}^{\text{S}}$ based on iteratively "pruning" those branches of $\Upsilon_{\mathcal{K}}^{\text{S}}$ that cannot be distinguished by $\mu\mathcal{L}_A^{\text{EQL}}$ properties.

In fact, $\Theta_{\mathcal{K}}^{\text{S}}$ is of size exponential in the size of the initial state of the S-KAB $\mathcal{K}$ and the bound b. Hence, considering the complexity of model checking of μ-calculus on finite-state transition systems [10, 22], we obtain that verification is in ExpTime.

4. Repair Semantics for KABs

S-KABs are defined by taking a radical approach in the management of inconsistency: simply reject actions that lead to T-inconsistent ABoxes. However, an inconsistency could be caused by a small portion of the ABox, making it desirable to *handle* the inconsistency by allowing the action execution, and taking care of *repairing* the resulting state so as to restore consistency while minimizing the information loss. To this aim, we revise the standard semantics for KABs so as to manage inconsistency, relying on the research on instance-level evolution of knowledge bases [24, 13, 16, 9], and, in particular, on the notion of *ABox repair*, cf. [3, 18].

In particular, we assume that in this case the system is equipped with a *repair service* that is executed every time an action changes the content of the ABox. In this light, a progression step of the KAB is constituted by two sub-steps: an *action step*, where an executable action with parameters is chosen and applied over the current ABox, followed by a *repair step*, where the repair service checks whether the resulting state is T-consistent or not, and, in the negative case, fixes the content of the ABox resulting from the action step, by applying its repair strategy.

Repairing ABoxes. We illustrate our approach by considering two specific forms of repair that have been proposed in the literature [13] and are applicable to the context of DL ontologies [18].

- Given an ABox A and a TBox T, a *bold-repair* (*b-repair*) of A with T is a maximal T-consistent subset A' of A. Clearly, there might be more than one bold-repair for given A and T. By $\textsc{rep}(A, T)$ we denote the set of *all* b-repairs of A with T.
- A *certain-repair* (*c-repair*) of A with T is the ABox defined as follows: $A' = \cap_{A'' \in \textsc{rep}(A,T)} A''$. That is, a c-repair of A with T contains only those ABox statements that occur in every b-repair of A with T.

In general, there are (exponentially) many b-repairs of an ABox A with T, while by definition there is a single c-repair.

Example 4.0.5. *Continuing Example 3.0.2, consider the T-inconsistent state $\langle A', \emptyset \rangle$ obtained from $\gamma_1()$ in A_0. Its two b-repairs are $\textsc{rep}(A', T) = \{A_1, A_2\}$ with $A_1 = \{C(a)\}$, $A_2 = \{D(a)\}$. Its c-repair is $\bigcap_{A \in \textsc{rep}(A', T)} A = \{C(a)\} \cap \{D(a)\} = \emptyset$.* □

4.1. Bold and Certain Repair Transition Systems

We now refine the execution semantics of KABs by constructing a two-layered transition system that reflects the alternation between the action and the repair steps. In particular, we consider the two cases for which the repair strategy either follows the bold or certain semantics. We observe that, if b-repair semantics is applied, then the repair service has,

in general, several possibilities to fix an inconsistent ABox. Since, a-priori, no information about the repair service can be assumed beside the repair strategy itself, the transition system capturing this execution semantics must consider the progression of the system for any computable repair, modelling the repair step as the result of a non-deterministic choice taken by the repair service when deciding which among the possible repairs will be the actually enforced one. This issue does not occur with c-repair semantics, because its repair strategy is deterministic.

In order to distinguish whether a state is obtained from an action or repair step, we introduce a special marker $\mathsf{State}(\mathit{temp})$, which is an ABox statement with a fresh concept name State and a fresh constant temp, s.t.: if $\mathsf{State}(\mathit{temp})$ is in the current state, this means that the state has been produced by an action step, otherwise by the repair step.

Formally, the *b-transition system* $\Upsilon^b_{\mathcal{K}}$ (resp. *c-transition system* $\Upsilon^c_{\mathcal{K}}$) for a KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$ is a (possibly infinite-state) transition system $(\mathcal{C}, T, \Sigma, s_0, \mathit{abox}, \Rightarrow)$ where:

(1) $s_0 = \langle A_0, \emptyset \rangle$;

(2) Σ and $\Rightarrow$ are defined by simultaneous induction as the smallest sets satisfying the following properties:

  (i) $s_0 \in \Sigma$;

  (ii) (*action step*) if $\langle A, m \rangle \in \Sigma$ and $\mathsf{State}(\mathit{temp}) \notin A$, then for all actions γ in Γ, for all substitutions σ for the parameters of γ and for all $\langle A', m' \rangle$ s.t. $\langle \langle A, m \rangle, \gamma\sigma, \langle A', m' \rangle \rangle \in \textsc{exec}_{\mathcal{K}}$, let $A'' = A' \cup \{\mathsf{State}(\mathit{temp})\}$, and then $\langle A'', m' \rangle \in \Sigma$ and $\langle A, m \rangle \Rightarrow \langle A'', m' \rangle$;

  (iii) (*repair step*) if $\langle A, m \rangle \in \Sigma$ and $\mathsf{State}(\mathit{temp}) \in A$, then for b-repair A' (resp. c-repair A') of $A - \{\mathsf{State}(\mathit{temp})\}$ with T, we have $\langle A', m \rangle \in \Sigma$ and $\langle A, m \rangle \Rightarrow \langle A', m \rangle$.

We refer to KABs with b-transition (resp. c-transition) system semantics as b-KAB (resp. c-KAB).

Example 4.1.1. *Under b-repair semantics, the KAB in our running example looks as follows. Since A' is T-inconsistent, we have two bold repairs, A_1 and A_2, which in turn give rise to two runs: $\langle A_0, \emptyset \rangle \Rightarrow \langle A'_r, \emptyset \rangle \Rightarrow \langle A_1, \emptyset \rangle$ and $\langle A_0, \emptyset \rangle \Rightarrow \langle A'_r, \emptyset \rangle \Rightarrow \langle A_2, \emptyset \rangle$, where $A'_r = \{A' \cup \{\mathsf{State}(\mathit{temp})\}$. Since instead γ_1 does not lead to any inconsistency, for every candidate successor $A'' = \{G(x)\}$ with $m = \{(f(a) \mapsto x)\}$ (see Example 3.0.3), we have $\langle A_0, \emptyset \rangle \Rightarrow \langle A'' \cup \{\mathsf{State}(\mathit{temp})\}, m \rangle \Rightarrow \langle A'', m \rangle$, reflecting that in this case the repair service just maintains the resulting ABox unaltered.* □

4.2. Verification Under Repair Semantics

We observe that the alternation between an action and a repair step makes EQL queries meaningless for the intermediate states produced as a result of action steps, because the resulting ABox could be in fact T-inconsistent (see, e.g., state $\langle A'_r, \emptyset \rangle$ in Example 4.1.1). In fact, such intermediate states are needed just to capture the dynamic structure that reflects the behaviour of the system. E.g., state $\langle A'_r, \emptyset \rangle$ in Example 4.1.1 has two successor states, attesting that the repair service with bold semantics will produce one between two possible repairs.

In this light, we introduce the *inconsistency-tolerant* temporal logic $\mu\mathcal{L}_A^{\mathrm{IT}}$, which is a fragment of $\mu\mathcal{L}_A^{\mathrm{EQL}}$ defined as:

$$\Phi \ := \ Q \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \exists x.\Phi \mid \langle\!\!-\!\!\rangle[\!\!-\!\!]\Phi \mid [\!\!-\!\!][\!\!-\!\!]\Phi \mid Z \mid \mu Z.\Phi$$

Beside the standard abbreviations introduced for $\mu\mathcal{L}_A^{\mathrm{EQL}}$, we also make use of the following: $\langle\!\!-\!\!\rangle\langle\!\!-\!\!\rangle\Phi = \neg[\!\!-\!\!][\!\!-\!\!]\neg\Phi$, and $[\!\!-\!\!]\langle\!\!-\!\!\rangle\Phi = \neg\langle\!\!-\!\!\rangle[\!\!-\!\!]\neg\Phi$. This logic can be used to express interesting properties over b- and c-KABs, exploiting different combinations of the $\langle\!\!-\!\!\rangle$ and $[\!\!-\!\!]$ next-state operators so as to quantify over the possible action steps and corresponding repair steps, ensuring at the same time that only the T-consistent states produced by the repair steps are queried. For example, $\mu Z.(\Phi \vee \langle\!\!-\!\!\rangle\langle\!\!-\!\!\rangle Z)$ models the "optimistic" reachability of Φ, stating that there exists a sequence of action and repair steps, s.t. Φ eventually holds. Conversely, $\mu Z.(\Phi \vee \langle\!\!-\!\!\rangle[\!\!-\!\!]Z)$ models the "robust" reachability of Φ, stating the existence of a sequence of action steps leading to Φ independently from the behaviour of the repair service. This patterns can be nested into more complex properties that express requirements about the acceptable progressions of the system, taking into account data and repairs. E.g., $\nu Z.(\forall x.Stud(x) \rightarrow \mu Y.(Grad(x) \vee \langle\!\!-\!\!\rangle[\!\!-\!\!]Y)) \wedge [\!\!-\!\!][\!\!-\!\!]Z$ states that, for every student x encountered in any state of the system, it is possible to "robustly" reach a state where x becomes graduated.

Since for a given ABox there exist finitely many b-repairs, and one c-repair, the technique used to prove decidability of properties for run-bounded S-KABs can be extended to deal with b- and c-KABs as well.

Theorem 4.2.1. *Verification of $\mu\mathcal{L}_A^{\mathrm{IT}}$ properties over run-bounded b-KABs and c-KABs is decidable.*

The precise relationship between b-KABs and c-KABs remains to be investigated.

5. Extended Repair Semantic for KABs

B-KABs and c-KABs provide an inconsistency-handling semantics to KABs. However, despite dealing with possible repairs when some action step produces a T-inconsistent ABox, they do not explicitly track whether a repair has been actually enforced, nor do they provide finer-grained insights about which TBox assertions were involved in the inconsistency. We extend the repair execution semantics of so as to equip the transition system with this additional information.

While *DL-Lite*$_\mathcal{A}$ does not allow, in general, to uniquely extract from a T-inconsistent ABox a set of individuals that are responsible for the inconsistency [7], its *separability* property [7] guarantees that inconsistency arises because a single negative TBox assertion is violated. More specifically, a T-inconsistency involves the violation of either a functionality assertion or negative inclusion in T. Since *DL-Lite*$_\mathcal{A}$ obeys to the restriction that no functional role can be specialized, the first case can be detected by just considering the ABox and the functionality assertion alone. Contrariwise, the second requires also to take into account the positive inclusion assertions (since disjointness propagates downward to the subclasses). Thanks to the FO rewritability of ontology satisfiability in *DL-Lite*$_\mathcal{A}$ [7], check can be done by constructing a FOL boolean query that corresponds to the considered functional or negative inclusion assertion, and that can be directly evaluated over the ABox, considered as a database of facts.

Following [7], given a functionality assertion ($\mathsf{funct}\ R$), we construct the query $q^f_{\text{unsat}}((\mathsf{funct}\ R)) = \exists x, x_1, x_2.\eta(R, x, x_1) \wedge \eta(R, x, x_2) \wedge x_1 \neq x_2$, where $\eta(R, x, y) = P(x, y)$ if $R = P$, and $\eta(R, x, y) = P(y, x)$ if $R = P^-$. Given a negative concept inclusion $B_1 \sqsubseteq \neg B_2$ and a set of positive inclusions T_p, we construct the query $q^n_{\text{unsat}}(B_1 \sqsubseteq \neg B_2, T_p) = \mathit{rew}(T_p, \exists x.\gamma(B_1, x) \wedge \gamma(B_2, x))$, where $\gamma(B, x) = N(x)$ if $B = N$, $\gamma(B, x) = P(x, \_)$ if $B = \exists P$, and $\gamma(B, x) = P(\_, x)$ if $B = \exists P^-$. Similarly, given a negative role inclusion $R_1 \sqsubseteq \neg R_2$, we construct the query $q^n_{\text{unsat}}(R_1 \sqsubseteq \neg R_2, T_p) = \mathit{rew}(T_p, \exists x_1, x_2.\eta(R_1, x_1, x_2) \wedge \eta(R_2, x_1, x_2))$.

5.1. Extended Repair Transition System

With this machinery at hand, given a KB (T, A) we can now compute the set of TBox assertions in T that are actually violated by A. To do so, we assume wlog that $\mathcal{C}_0$ contains one distinguished constant per TBox assertion in T, and introduce a function LABEL, that, given a TBox assertion, returns the corresponding constant. We then define the set VIOL(A, T) of constants labeling TBox assertions in T violated by A, as:

$$\{d \in \Delta \mid \exists t \in T_f \text{ s.t. } d = \text{LABEL}(t) \text{ and } A \models q^f_{\text{unsat}}(t)\} \cup$$
$$\{d \in \Delta \mid \exists t \in T_n \text{ s.t. } d = \text{LABEL}(t) \text{ and } A \models q^n_{\text{unsat}}(t, T_p)\}$$

Example 5.1.1. *Consider $\mathcal{K}$ in Example 2.2.1, with $T = \{C \sqsubseteq \neg D\}$, and $A' = \{D(a), C(a)\}$ in Example 3.0.2. Assume that* $\textsc{label}(C \sqsubseteq \neg D) = \ell$. *We have $\phi = q^n_{\text{unsat}}(C \sqsubseteq \neg D, \emptyset) = \exists x.C(x) \wedge D(x)$. Since $A' \models \phi$, we get* $\textsc{viol}(A', T) = \{\ell\}$.

We now employ this information assuming that the repair service decorates the states it produces with information about which TBox functional and negative inclusion assertions have been involved in the repair. This is done with a fresh concept Viol that keeps track of the labels of violated TBox assertions.

Formally, we define the *eb-transition system* $\Upsilon^{eb}_{\mathcal{K}}$ (resp. *ec-transition system* $\Upsilon^{ec}_{\mathcal{K}}$) for KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$ as a (possibly infinite-state) transition system $(\mathcal{C}, T, \Sigma, s_0, \mathit{abox}, \Rightarrow)$ constructed starting from $\Upsilon^{b}_{\mathcal{K}}$ (resp. $\Upsilon^{c}_{\mathcal{K}}$) by refining the repair step as follows: if $\langle A, m\rangle \in \Sigma$ and $\mathsf{State}(\mathit{temp}) \in A$, then for b-repair A' (resp. c-repair A') of $A - \{\mathsf{State}(\mathit{temp})\}$ with T, we have $\langle A'_v, m\rangle \in \Sigma$ and $\langle A, m\rangle \Rightarrow \langle A'_v, m\rangle$, where $A'_v = A' \cup \{\mathsf{Viol}(d) \mid d \in \textsc{viol}(A', T)\}$.

5.2. Verification Under Extended Repair Semantics

Thanks to the insertion of information about violated TBox assertions in their transition systems, eb-KABs and ec-KABs support the verification of $\mu\mathcal{L}^{\text{IT}}_A$ properties that mix dynamic requirements with queries over the instance-level information and over the meta-level information related to inconsistency. Notice that such properties can indirectly refer to specific TBox assertions, thanks to the fact that their labels belong to the set of distinguished constants $\mathcal{C}_0$. Examples of formulae focused on the presence of violations in the system are:

- $\nu Z.(\neg\exists l.\mathsf{Viol}(l)) \wedge [\!-\!][\!-\!]Z$ says that no state of the system is manipulated by the repair service;
- $\nu Z.(\forall l.\mathsf{Viol}(l) \rightarrow (\mu Y.\nu W.\neg\mathsf{Viol}(l) \wedge [\!-\!][\!-\!]W \vee \langle\!-\!\rangle[\!-\!]Y) \wedge [\!-\!][\!-\!]Z$ says that, in all states, whenever a TBox assertion a is violated, independently from the applied repairs there exists a run that reaches a state starting from which a will never be violated anymore.

Since the TBox assertions are finitely many and fixed for a given KAB, the key decidability result of Theorem 4.2.1 can be seamlessly carried over to these extended repair semantics.

Theorem 5.2.1. *Verification of $\mu\mathcal{L}^{\text{IT}}_A$ properties over run-bounded eb-KABs and ec-KABs is decidable.*

5.3. From Standard to Extended Repair KABs

It is clear that extended repair KABs are richer than repair KABs. We now show that eb- and ec-KABs are also richer than S-KABs, thanks to the fact that information about the violated TBox assertions is explicitly tracked in all states resulting from a repair step. In particular, verification of $\mu\mathcal{L}^{\text{EQL}}_A$ properties over a KAB $\mathcal{K}$ under standard semantics can be recast as a corresponding verification problem over $\mathcal{K}$ interpreted either under

extended bold or extended certain repair semantics. The intuition behind the reduction is that a property holds over $\Upsilon^s_{\mathcal{K}}$ if that property holds in the portion of the $\Upsilon^{eb}_{\mathcal{K}}$ (or $\Upsilon^{ec}_{\mathcal{K}}$) where no TBox assertion is violated. The absence of violation can be checked over T-consistent states by issuing the EQL query $\neg\exists x.[\mathsf{Viol}(x)]$. Technically, we define a translation function τ that transforms an arbitrary $\mu\mathcal{L}_A^{\text{EQL}}$ property Φ into a $\mu\mathcal{L}_A^{\text{IT}}$ property $\Phi' = \tau(\Phi)$. The translation $\tau(\Phi)$ is inductively defined by recurring over the structure of Φ and substituting each occurrence of $\langle\!-\!\rangle\Psi$ with $\langle\!-\!\rangle\langle\!-\!\rangle((\neg\exists x.\mathsf{Viol}(x)) \wedge \tau(\Psi))$, and each occurrence of $[\!-\!]\Psi$ with $[\!-\!]\langle\!-\!\rangle((\neg\exists x.\mathsf{Viol}(x)) \rightarrow \tau(\Psi))$. Observe that, in τ, the choice of $\langle\!-\!\rangle$ for the nested operator can be substituted by $[\!-\!]$, because for T-consistent states produced by an action step, the repair step simply copy the resulting state, generating a unique successor even in the eb-semantics.

Theorem 5.3.1. *Given a KAB $\mathcal{K}$ and a $\mu\mathcal{L}_A^{\text{EQL}}$ property Φ, $\Upsilon^s_{\mathcal{K}} \models \Phi$ iff $\Upsilon^{eb}_{\mathcal{K}} \models \tau(\Phi)$ iff $\Upsilon^{ec}_{\mathcal{K}} \models \tau(\Phi)$.*

The correctness of this result can be directly obtained by considering the semantics of $\mu\mathcal{L}_A^{\text{EQL}}$ and $\mu\mathcal{L}_A^{\text{IT}}$, and the construction of the transition systems under the three semantics.

6. Weakly Acyclic KABs

So far, all the decidability results here presented have relied on the assumption that the considered KAB is state-bounded. As pointed out in [2], *run boundedness* is a semantic condition that is undecidable to check. In [2], a sufficient, syntactic condition borrowed from *weak acyclicity* in data exchange [15] has been proposed to actually check whether the KAB under study is run bounded and, in turn, verifiable.

Intuitively, given a KAB $\mathcal{K}$, this test constructs a dependency graph tracking how the actions of $\mathcal{K}$ transport values from one state to the next one. To track all the actual dependencies, every involved query is first rewritten considering the positive inclusion assertions of the TBox. Two types of dependencies are tracked: copy of values and use of values as parameters of a service call. $\mathcal{K}$ is said to be *weakly acyclic* if there is no cyclic chain of dependencies of the second kind. The presence of such a cycle could produce an infinite chain of fresh values generation through service calls.

The crux of the proof showing that weakly acyclicity ensures run boundedness is based on the notion of *positive dominant*, which creates a simplified version of the KAB that, from the execution point of view, obeys to three key properties. First, its execution consists of a single run that closely resembles the chase of a set of tuple-generating dependencies, which terminates under the assumption of weak acyclicity [15], guaranteeing that the positive dominant is indeed run-bounded. Second, it considers only the positive inclusion assertions of the TBox, therefore producing always the same behaviour independently from which execution semantics is chosen, among the ones discussed in this paper. Third, for every run contained in each of the transition systems generated under the standard, bold repair, certain repair, and their extended versions, the values accumulated along the run are "bounded" by the ones contained in the unique run of the positive dominant. This makes it possible to directly carry run-boundedness from the positive dominant to the original KAB, independently from which execution semantics is considered.

Theorem 6.0.2. *Given a weakly acyclic KAB $\mathcal{K}$, we have that $\Upsilon^{s}_{\mathcal{K}}$, $\Upsilon^{b}_{\mathcal{K}}$, $\Upsilon^{c}_{\mathcal{K}}$, $\Upsilon^{eb}_{\mathcal{K}}$, $\Upsilon^{ec}_{\mathcal{K}}$ are all run-bounded.*

Theorem 6.0.2 shows that weak acyclicity is an effective method to check verifiability of KABs under all inconsistency-aware semantics considered in this paper.

7. Conclusion

We have approached the problem of inconsistency handling in Knowledge and Action Bases, by resorting to an approach based on ABox repairs. An orthogonal approach to the one taken is to maintain ABoxes that are inconsistent with the TBox as states of the transition system, and rely, both for the progression mechanism and for answering queries used in verification, on consistent query answering [3, 18]. Notably, we are able to show that the decidability and complexity results established for the repair-based approaches carry over also to this setting. It remains open to investigate the relationship between these orthogonal approaches to dealing with inconsistency in KABs.

Bibliography

[1] Babak Bagheri Hariri, Diego Calvanese, Giuseppe De Giacomo, Riccardo De Masellis, Marco Montali, and Paolo Felli. Verification of description logic Knowledge and Action Bases. In *Proc. of the 20th Eur. Conf. on Artificial Intelligence (ECAI 2012)*, pages 103–108, 2012.

[2] Babak Bagheri Hariri, Diego Calvanese, Giuseppe De Giacomo, Alin Deutsch, and Marco Montali. Verification of relational data-centric dynamic systems with external services. In *Proc. of the 32nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2013)*, 2013.

[3] Leopoldo E. Bertossi. Consistent query answering in databases. *SIGMOD Record*, 35(2):68–76, 2006.

[4] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification of infinite structures. In *Handbook of Process Algebra*. Elsevier Science, 2001.

[5] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodríguez-Muro, and Riccardo Rosati. Ontologies and databases: The *DL-Lite* approach. In Sergio Tessaris and Enrico Franconi, editors, *Reasoning Web. Semantic Technologies for Informations Systems – 5th Int. Summer School Tutorial Lectures (RW 2009)*, volume 5689 of *Lecture Notes in Computer Science*, pages 255–356. Springer, 2009.

[6] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. EQL-Lite: Effective first-order query processing in description logics. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007)*, pages 274–279, 2007.

[7] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.

[8] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Marco Montali, and Ario Santoso. Ontology-based governance of data-aware processes. In *Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems (RR 2012)*, volume 7497 of *Lecture Notes in Computer Science*, pages 25–41. Springer, 2012.

[9] Diego Calvanese, Evgeny Kharlamov, Werner Nutt, and Dmitriy Zheleznyakov. Evolution of *DL-Lite* knowledge bases. In *Proc. of the 9th Int. Semantic Web Conf.*

(ISWC 2010), volume 6496 of *Lecture Notes in Computer Science*, pages 112–128. Springer, 2010.

[10] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model checking*. The MIT Press, Cambridge, MA, USA, 1999.

[11] Alin Deutsch, Richard Hull, Fabio Patrizi, and Victor Vianu. Automatic verification of data-centric business processes. In *Proc. of the 12th Int. Conf. on Database Theory (ICDT 2009)*, pages 252–267, 2009.

[12] Alin Deutsch, Liying Sui, and Victor Vianu. Specification and verification of data-driven web applications. *J. of Computer and System Sciences*, 73(3):442–474, 2007.

[13] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.

[14] E. Allen Emerson. Model checking and the Mu-calculus. In N. Immerman and P. Kolaitis, editors, *Proc. of the DIMACS Symposium on Descriptive Complexity and Finite Model*, pages 185–214. American Mathematical Society Press, 1997.

[15] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[16] Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis, and Grigoris Antoniou. Ontology change: Classification and survey. *Knowledge Engineering Review*, 23(2):117–152, 2008.

[17] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning – Theory and Practice*. Elsevier, 2004.

[18] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Inconsistency-tolerant semantics for description logics. In *Proc. of the 4th Int. Conf. on Web Reasoning and Rule Systems (RR 2010)*, pages 103–117, 2010.

[19] Lior Limonad, Pieter De Leenheer, Mark Linehan, Rick Hull, and Roman Vaculin. Ontology of dynamic entities. In *Proc. of the 31st Int. Conf. on Conceptual Modeling (ER 2012)*, 2012.

[20] David Michael Ritchie Park. Finiteness is Mu-ineffable. *Theoretical Computer Science*, 3(2):173–181, 1976.

[21] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.

[22] Colin Stirling. *Modal and Temporal Properties of Processes*. Springer, 2001.

[23] Victor Vianu. Automatic verification of database-driven systems: a new frontier. In *Proc. of the 12th Int. Conf. on Database Theory (ICDT 2009)*, pages 1–13, 2009.

[24] Marianne Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.

A. Bisimulation and Invariance

We start by introducing the notion of isomorphism between ABoxes. Two ABoxes A_1 and A_2 are *isomorphic*, written $A_1 \equiv A_2$, if there exists a bijection $h : S_1 \to S_2$, with $\textsc{adom}(A_1) \cup \mathcal{C}_0 \subseteq S_1$ and $\textsc{adom}(A_2) \cup \mathcal{C}_0 \subseteq S_2$, which is the identity on $\mathcal{C}_0$, and s.t.:

1. for every concept assertion $N(d) \in A_1$, $N(h(d)) \in A_2$;
2. for every role assertion $P(d_1, d_2) \in A_1$, $N(h(d_1), h(d_2)) \in A_2$;
3. for every concept assertion $N(d) \in A_2$, $N(h^{-1}(d)) \in A_1$;
4. for every role assertion $P(d_1, d_2) \in A_2$, $N(h^{-1}(d_1), h^{-1}(d_2)) \in A_1$.

We write $A_1 \equiv_h A_2$ to make h explicit. Furthermore, with a slight abuse of notation, we write $A_2 = h(A_1)$, and $A_1 = h^{-1}(A_2)$, when there exists a bijection $h : S_1 \to S_2$, with $\textsc{adom}(A_1) \cup \mathcal{C}_0 \subseteq S_1$ and $\textsc{adom}(A_2) \cup \mathcal{C}_0 \subseteq S_2$, s.t. $A_1 \equiv_h A_2$.

It is easy to see that isomorphism implies the following results.

Lemma A.0.3. *Consider two knowledge bases (T, A_1) and (T, A_2), s.t. there exists a bijection h with $A_2 = h(A_1)$. For every EQL query q, we have $\langle d_1, \ldots, d_n \rangle \in \textsc{ans}(q, T, A_1)$ iff $\langle h(d_1), \ldots, h(d_n) \rangle \in \textsc{ans}(h(q), T, h(A_1))$.*

Proof. Trivial, by recalling the notion of first-order rewritability of EQL queries, and the fact that first-order logic cannot distinguish between isomorphic structures. □

We now recast the notion of *history preserving bisimulation* as defined in [2] in the context of KABs. Let $\Upsilon_1 = (\mathcal{C}_1, T, \Sigma_1, s_0, abox_1, \Rightarrow_1)$ and $\Upsilon_1 = (\mathcal{C}_2, T, \Sigma_2, s_0, abox_2, \Rightarrow_2)$ be transition systems, with $abox(s_0) \subseteq \mathcal{C}_0 \subseteq \mathcal{C}_1 \cap \mathcal{C}_2$. Let H be the set of partial bijections between $\mathcal{C}_1$ and $\mathcal{C}_2$, which are the identity over $\mathcal{C}_0$. A *history preserving bisimulation* between Υ_1 and Υ_2 is a relation $\mathcal{B} \subseteq \Sigma_1 \times H \times \Sigma_2$ such that $\langle s_1, h, s_2 \rangle \in \mathcal{B}$ implies that:

1. h is a partial bijection between $\mathcal{C}_1$ and $\mathcal{C}_2$, s.t. h fixes $\mathcal{C}_0$ and $abox_1(s_1) \equiv_h abox_2(s_2)$;
2. for each s_1', if $s_1 \Rightarrow_1 s_1'$ then there is an s_2' with $s_2 \Rightarrow_2 s_2'$ and a bijection h' that extends h, such that $\langle s_1', h', s_2' \rangle \in \mathcal{B}$.
3. for each s_2', if $s_2 \Rightarrow_2 s_2'$ then there is an s_1' with $s_1 \Rightarrow_1 s_1'$ and a bijection h' that extends h, such that $\langle s_1', h', s_2' \rangle \in \mathcal{B}$.

A state $s_1 \in \Sigma_1$ is *history preserving bisimilar* to $s_2 \in \Sigma_2$ *wrt a partial bijection* h, written $s_1 \approx_h s_2$, if there exists a history preserving bisimulation $\mathcal{B}$ between Υ_1 and Υ_2 such that $\langle s_1, h, s_2 \rangle \in \mathcal{B}$. A state $s_1 \in \Sigma_1$ is *history preserving bisimilar* to $s_2 \in \Sigma_2$, written $s_1 \approx s_2$, if there exists a partial bijection h and a history preserving bisimulation $\mathcal{B}$ between Υ_1 and Υ_2 such that $\langle s_1, h, s_2 \rangle \in \mathcal{B}$. A transition system Υ_1 is *history preserving bisimilar* to Υ_2, written $\Upsilon_1 \approx \Upsilon_2$, if there exists a partial bijection h_0 and a history preserving bisimulation $\mathcal{B}$ between Υ_1 and Υ_2 such that $\langle s_{01}, h_0, s_{02} \rangle \in \mathcal{B}$.

The following fundamental results connects history preserving bisimulation and the logic $\mu\mathcal{L}_A^{\text{EQL}}$:

Theorem A.0.4. *Consider two transition systems* Υ_1 *and* Υ_2 *such that* $\Upsilon_1 \approx \Upsilon_2$*. For every* $\mu\mathcal{L}_A^{\mathrm{EQL}}$ *closed formula* Φ*, we have:* $\Upsilon_1 \models \Phi$ *if and only if* $\Upsilon_2 \models \Phi$*.*

Proof. The proof follows from that of Theorem 3.1 in [2], noticing that, by Lemma A.0.3, isomorphism indeed preserves certain answers. □

B. Standard KABs

B.1. Proof of Theorem 3.0.4

In principle, decidability can be obtained by taking advantage from first-order rewritability of *DL-Lite*$_{\mathcal{A}}$, and translating a KAB into a corresponding Data-Centric Dynamic System [2]. However, in order to make the proof adaptable to the inconsistency-aware semantics discussed in the paper, we reconstruct the proof contained in [2] over KABs. We first discuss the intuition behind the proof, and then focus on the technical development.

Given a run-bounded S-KAB $\mathcal{K}$, the crux of the proof is to show how to construct an *abstract transition system* $\Theta^{\mathrm{S}}_{\mathcal{K}}$ that satisfies exactly the same $\mu\mathcal{L}^{\mathrm{EQL}}_{A}$ properties of the original transition system $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$. To do so, a first observation is that the only source of infiniteness in $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$ is the infinite branching arising when a service call is issued for the first time. In this case, given a state $s = \langle A, m\rangle$ in $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$, for every executable action with legal parameters $\alpha\sigma$, s contains an infinite number of successor states, each one corresponding to an assignment of all the newly introduced service calls to values in $\mathcal{C}$, s.t. the resulting state does not violate any axiom of T.

One can see these successors as variations of a finite set of structures, each one expressing an isomorphic type (called *equality commitment*) constructed over the set of facts $E = \textsc{do}(T, A, \alpha\sigma)$ and the map m, by fixing the set of equalities and inequalities between the service calls that must be issued, and the service calls and values contained in E, m and $\mathcal{C}_0$. Each structure can be concretized into a successor state by evaluating the service calls so as to satisfy the equalities and inequalities induced by the equality commitment (this also guarantees that the evaluation agrees with m). Two concretizations of the same structure are isomorphic, i.e., they contain the same ABox and service call map modulo renaming of the newly introduced values.

We now observe that EQL-queries do not distinguish isomorphic ABoxes. In particular, consider two ABoxes A_1 and A_2, and a bijection h that induces an isomorphism between A_1 and A_2. Now consider an EQL query q s.t. the constants used in q appear in h, and let $h(q)$ be the query obtained by replacing such constants through the application of h. It is easy to see that the certain answers of q over A_1 are exactly the same of $h(q)$ over A_2, modulo renaming of the values via h. The key consequence of this property is that, given a state s of $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$, $\mu\mathcal{L}^{\mathrm{EQL}}_{A}$ is not able to distinguish successors of s that concretize E by satisfying the same equality commitment. Therefore, all such successors can be collapsed into a unique representative successor, without affecting the satisfaction of a closed $\mu\mathcal{L}^{\mathrm{EQL}}_{A}$ property Φ asked in the initial state of the system.

By inductively applying this pruning, we can construct a finite-state transition system $\Theta^{\mathrm{S}}_{\Phi}$. Since the active domain of $\Theta^{\mathrm{S}}_{\Phi}$ is finite, by quantifier elimination we can then transform Φ into a corresponding propositional μ-calculus property ϕ, and reduce verification

of Φ over $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$ as standard model checking of ϕ over $\Theta^{\mathrm{S}}_{\mathcal{K}}$, which is indeed decidable [14].

Equality commitments. Given a set $S \subseteq \mathbb{SC} \cup \mathcal{C}$ containing individuals and service calls, an *equality commitment* over S is a partition H of S s.t. every cell of H contains at most one element $d \in \mathcal{C}$. Given an element $e \in S$, we use $[e]_H$ do denote the cell e belongs to. With a slight abuse of notation, we say that $e \in H$ if $e \in S$. Now consider a KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$, a state $\langle A, m\rangle$, and an action $\alpha \in \Gamma$ with parameters σ, s.t. $\alpha\sigma$ is legal in $\langle A, m\rangle$ according to Π. Let $\mathcal{H}(T, \langle A, m\rangle, \alpha\sigma)$ be the set of equality commitments H_i constructed over $\textsc{adom}(\mathcal{C}_0) \cup \textsc{adom}(A) \cup \textsc{dom}(m) \cup \textsc{im}(m) \cup \textsc{adom}(\textsc{do}(T, A, \alpha\sigma))$ that *agrees* with m, i.e., for every assignment $(f \to d)$ in m, $[f]_{H_i} = [d]_{H_i}$. Intuitively, the elements of $\mathcal{H}$ are equality commitments that fix the equivalence class to which every new service call, introduced by $\textsc{do}(T, A, \alpha\sigma)$, belongs to.

We say that $\textsc{evals}(T, A, \alpha\sigma)$ *respects* an equality commitment $H \in \mathcal{H}(T, \langle A, m\rangle, \alpha\sigma)$ if, for every pair of assignments $(f_1 \to d_1), (f_2 \to d_2)$ in $\textsc{evals}(T, A, \alpha\sigma)$, $d_1 = d_2$ iff f_1 and f_2 belong to the same cell P of H, and $d_1 = d_2 = d$ iff d belongs to P.

Pruning. Given a KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$, we refine the definition of $\textsc{exec}_{\mathcal{K}}$ so as to create a parsimonious version that minimally covers, state-by-state, the various equality commitments.

In particular, we define a transition relation $\textsc{p-exec}_{\mathcal{K}}$ as follows. For every $\langle\langle A, m\rangle, \alpha\sigma, \langle A', m'\rangle\rangle \in \textsc{exec}_{\mathcal{K}}$, fix $\theta = m' \setminus m$ and $H \in \mathcal{H}(T, \langle A, m\rangle, \alpha\sigma)$ s.t. θ respects H. Then there exists *only one* $\theta_p = \textsc{evals}(T, A, \alpha\sigma)$ s.t. θ_p respects H and, given, $A_p = \textsc{do}(T, A, \alpha\sigma)\theta_p$ and $m' = m \cup \theta_p$, $\langle\langle A, m\rangle, \alpha\sigma, \langle A_p, m_p\rangle\rangle \in \textsc{p-exec}_{\mathcal{K}}$. Intuitively, $\textsc{p-exec}_{\mathcal{K}}$ "prunes" $\textsc{exec}_{\mathcal{K}}$ by collapsing into a unique representative tuple all transitions that are associated to a given starting state and action with parameters, and that respect the same equality commitment.

Starting from $\textsc{p-exec}_{\mathcal{K}}$, we define a *pruning* $\Theta^{\mathrm{S}}_{\mathcal{K}}$ of the transition system under standard semantics $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$ as a transition system constructed following the standard semantics, but by using $\textsc{p-exec}_{\mathcal{K}}$ in place of $\textsc{exec}_{\mathcal{K}}$ to inductively construct the set of states and transitions. In general, there exist infinitely many prunings, whose difference relies in the particular choice for the representatives when constructing $\textsc{p-exec}_{\mathcal{K}}$. However, we show that all such prunings are history-preserving bisimilar to the original transition system $\Theta_{\mathcal{K}}$. The following lemma is a key result in this direction, and intuitively shows that bisimulation does not distinguish different progressions that fix, step-by-step, the same equality commitments. In the lemma, for the sake of readability, given a service call map m and a function $h : \mathcal{C} \to \mathcal{C}$ defined over all values contained in m_1 (considering both the service call parameters and their results), we write $m_2 = f(m_1)$ to denote the service call map constructed as follows: for every assignment $(f(d_1, \ldots, d_n) \to d)$ in m_1, we have $(f(h(d_1), \ldots, h(d_n)) \to h(d))$ in m_2.

Lemma B.1.1. *Let $\mathcal{K}$ be a S-KAB with transition system $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$, and let $\Theta^{\mathrm{S}}_{\mathcal{K}}$ be a pruning of $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$. Consider a state $\langle A, m\rangle$ of $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$ and a state $\langle A_p, m_p\rangle$ of $\Theta^{\mathrm{S}}_{\mathcal{K}}$. If there exists a bijection h s.t. $A_p = h(A)$ and $m_p = h(m)$ (or, equivalently, $m = h^{-1}(m_p)$), then $\langle A, m\rangle \approx_h \langle A_p, m_p\rangle$.*

Proof. Let $\mathcal{K} = (T, A_0, \Gamma, \Pi)$, $\Upsilon^{\mathrm{S}}_{\mathcal{K}} = (\mathcal{C}, T, \Sigma, s_0, abox, \Rightarrow)$, and $\Theta^{\mathrm{S}}_{\mathcal{K}} = (\mathcal{C}, T, \Sigma_p, s_0, abox, \Rightarrow_p)$. To prove the lemma, we show that, for every state $\langle A', m'\rangle$

s.t. $\langle A, m\rangle \Rightarrow \langle A', m'\rangle$, there exists a state $\langle A'_p, m'_p\rangle$ and a bijection h' s.t.: 1. $\langle A_p, m_p\rangle \Rightarrow_p \langle A'_p, m'_p\rangle$; 2. h' extends h; 3. $A'_p = h'(A')$; 4. $m'_p = h'(m')$. By definition of Υ^{S}_K, if $\langle A, m\rangle \Rightarrow \langle A', m'\rangle$, then there exists an action $\alpha \in \Gamma$ with parameters σ s.t. σ is legal in $\langle A, m\rangle$ according to Π, and $\theta \in \textsc{evals}(T, A, \alpha\sigma)$ s.t. θ agrees with m, $A' = \textsc{do}(T, A, \alpha\sigma)\theta$, and $m' = m \cup \theta$. From this information, we can extract the equality commitment $H \in \mathcal{H}(T, \langle A, m\rangle, \alpha\sigma)$ s.t. θ respects H.

Since $A_p = h(A)$, from Lemma A.0.3 we know that the certain answers computed over A are the same, modulo renaming through h, to those computed over A_p. Furthermore, since σ maps parameters of α to values in $\textsc{adom}(A)$, we can construct σ_p mapping parameters of α to values in $\textsc{adom}(A_p)$, so as $(x \to d)$ in σ implies $(x \to h(d))$ in σ_p. By hypothesis, we also know that $m_p = h(m)$. As a consequence, we have that σ_p is legal in $\langle A_p, m_p\rangle$ according to Π, and that $\mathcal{H}(T, \langle A_p, m_p\rangle, \alpha\sigma_p)$ contains the same equality commitments in $\mathcal{H}(T, \langle A, m\rangle, \alpha\sigma)$ up to renaming of individuals through h. Now pick commitment $H_p \in \mathcal{H}(T, \langle A_p, m_p\rangle, \alpha\sigma_p)$ so that H_p corresponds to H up to renaming of individuals through h.

By definition of pruning, we know that there exists a unique θ_p that respects H_p (and, in turn, agrees with m_p) s.t., given $A'_p = \textsc{do}(T, A_p, \alpha\sigma_p)\theta_p$ and $m'_p = m_p \cup \theta_p$, we have $\langle A_p, m_p\rangle \Rightarrow_p \langle A'_p, m'_p\rangle$. Since H_p corresponds to H up to renaming of individuals through h, θ respects H, and θ_p respects H_p, we can lift h to an extended bijection h' s.t. $\theta_p = h(\theta)$. By construction, this means that $A'_p = h'(A')$, and that $m'_p = h'(m')$, hence the claim is proven.

The other direction can be proven in the symmetric way. □

Lemma B.1.2. *For every S-KAB $\mathcal{K}$ with transition system $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$ and every pruning $\Theta^{\mathrm{S}}_{\mathcal{K}}$ of $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$, we have $\Theta^{\mathrm{S}}_{\mathcal{K}} \approx \Upsilon^{\mathrm{S}}_{\mathcal{K}}$.*

Proof. Immediate consequence of Lemma B.1.1, by noticing that the initial states of $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$ and $\Theta^{\mathrm{S}}_{\mathcal{K}}$ are the same, and can be therefore connected through the identity bijection between their active domains. □

Proof of Theorem 3.0.4. Given a run-bounded KAB $\mathcal{K}$, we observe that each pruning $\Theta^{\mathrm{S}}_{\mathcal{K}}$ of $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$ is finite-state. On the one hand, thanks to run-boundedness each run consists of a finite number of states. On the other hand, thanks to the definition of pruning, each state has only finitely many successors. In fact, given a state of $\Theta^{\mathrm{S}}_{\mathcal{K}}$, there are only finitely many equality commitments that can be created by considering all possible actions with parameters. This implies that the entire active domain $\textsc{adom}(\Theta^{\mathrm{S}}_K)$ of $\Theta^{\mathrm{S}}_{\mathcal{K}}$ is finite as well. By Lemma B.1.2 and Theorem A.0.4, we know that $\Theta^{\mathrm{S}}_{\mathcal{K}}$ is a faithful abstraction of $\Upsilon^{\mathrm{S}}_{\mathcal{K}}$, i.e., for every $\mu\mathcal{L}^{\mathrm{EQL}}_A$ formula Φ, $\Upsilon^{\mathrm{S}}_{\mathcal{K}} \models \Phi$ iff $\Theta^{\mathrm{S}}_{\mathcal{K}} \models \Phi$. Taking advantage from the finiteness of $\textsc{adom}(\Theta^{\mathrm{S}}_K)$, by quantifier elimination we can construct a propositional μ-calculus property ϕ s.t. $\Theta^{\mathrm{S}}_{\mathcal{K}} \models \Phi$ iff $\Theta^{\mathrm{S}}_{\mathcal{K}} \models \phi$. The proof completes by observing that verifying whether $\Theta^{\mathrm{S}}_{\mathcal{K}} \models \phi$ amounts to standard model checking of propositional μ-calculus over finite-state transition systems, which is indeed decidable [14].

C. KABs Under Repair Semantics

We open this section by observing that the repair service does not distinguish between isomorphic ABoxes.

Lemma C.0.3. *Consider two knowledge bases (T, A_1) and (T, A_2), s.t. there exists a bijection h with $A_2 = h(A_1)$. Then for every ABox A_1^r s.t. $A_1^r \in \text{REP}(A_1, T)$, we have $h(A_2^r) \in \text{REP}(A_2, T)$, and for every ABox A_2^r s.t. $A_2^r \in \text{REP}(A_2, T)$, we have $h^{-1}(A_2^r) \in \text{REP}(A_1, T)$.*

Proof. Trivial, by recalling the notion of first-order rewritability of ontology satisfiability in *DL-Lite*$_\mathcal{A}$, and the fact that first-order logic cannot distinguish between isomorphic structures. □

C.1. Proof of Theorem 4.2.1

Given a $\mathcal{K}$, we introduce the *pruning* $\Theta_\mathcal{K}$ of the transition system under repair semantics (denoted by $\Upsilon^b_\mathcal{K}$ for the bold semantics, and $\Upsilon^c_\mathcal{K}$ for the certain semantics), as the transition system constructed following one between the two repair semantics, but by relying on the transition relation $\text{P-EXEC}_\mathcal{K}$ (as defined in Section B.1) in place of $\text{EXEC}_\mathcal{K}$. Differently from the standard case, to show that $\Theta_\mathcal{K} \approx \Upsilon^b_\mathcal{K}$ ($\Theta_\mathcal{K} \approx \Upsilon^c_\mathcal{K}$ resp.) we have to deal with the action and repair step. In particular, we reconstruct Lemma B.1.1 in this two-steps setting.

Lemma C.1.1. *Let $\mathcal{K}$ be a b-KAB (c-KAB respectively) with transition system $\Upsilon^b_\mathcal{K}$ ($\Upsilon^c_\mathcal{K}$ resp.), and let $\Theta_\mathcal{K}$ be a pruning of $\Upsilon^b_\mathcal{K}$ ($\Upsilon^c_\mathcal{K}$ resp.). Consider a state $\langle A, m\rangle$ of $\Upsilon^b_\mathcal{K}$ ($\Upsilon^c_\mathcal{K}$ resp.), and a state $\langle A_p, m_p\rangle$ of $\Theta_\mathcal{K}$. If there exists a bijection h s.t. $A_p = h(A)$ and $m_p = h(m)$ (or, equivalently, $m = h^{-1}(m_p)$), then $\langle A, m\rangle \approx_h \langle A_p, m_p\rangle$.*

Proof. Let $\mathcal{K} = (T, A_0, \Gamma, \Pi)$, $\Upsilon^b_\mathcal{K} = (\mathcal{C}, T, \Sigma, s_0, abox, \Rightarrow)$ (resp., $\Upsilon^c_\mathcal{K} = (\mathcal{C}, T, \Sigma, s_0, abox, \Rightarrow)$), and $\Theta_\mathcal{K} = (\mathcal{C}, T, \Sigma_p, s_0, abox, \Rightarrow_p)$. To prove the lemma, we show that, for every state $\langle A', m'\rangle$ s.t. $\langle A, m\rangle \Rightarrow \langle A', m'\rangle$, there exists a state $\langle A'_p, m'_p\rangle$ and a bijection h' s.t.: 1. $\langle A_p, m_p\rangle \Rightarrow_p \langle A'_p, m'_p\rangle$; 2. h' extends h; 3. $A'_p = h'(A')$; 4. $m'_p = h'(m')$. To show the claim, we have to separately discuss the case in which $\mathsf{State}(temp) \notin A$, and the case in which $\mathsf{State}(temp) \in A$. The first case is equivalent for $\Upsilon^b_\mathcal{K}$ and $\Upsilon^c_\mathcal{K}$, whereas the second case is different, since the two semantics diverge when it comes to the repair step (b-KABs nondeterministically produce one among the possible repairs, while c-KABs construct a unique repair corresponding to the intersection of possible repairs).

Base case: trivial, because the transition system and its pruning start from the same intial state $\langle A_0, \emptyset \rangle$.

Case 1 (action step): $\mathsf{State}(temp) \notin A$. First of all, we observe that *temp* is a distinguished constant of $\mathcal{C}_0$, hence $h(rep) = rep$. Since $A \equiv_h A_p$, $\mathsf{State}(temp) \notin A_p$. The claim can be then proven exactly in the same way as done for Lemma B.1.1, noticing however that each ABox A' s.t. $A \Rightarrow A'$ contains $\mathsf{State}(temp)$, making the induction hypothesis for case 1 inapplicable, and the one for case 2 applicable.

Case 2 (repair step) - bold semantics: $\mathsf{State}(temp) \in A$. By hypothesis, $A_p = h(A)$, and since $h(rep)$, $\mathsf{State}(temp) \in A_p$ as well. Notice that h is syntactically applied over the ABoxes A and A_p without involving the TBox T, and therefore it can be applied also when such ABoxes are T-inconsistent. On the one hand, by construction of the transition system under the bold repair semantics, we therefore know that:

1. for every s' s.t. $\langle A, m \rangle \Rightarrow s'$, we have $s' = \langle A', m \rangle$, with $A' \in \textsc{rep}(A - \{\mathsf{State}(temp)\}, T)$;
2. for every s'_p s.t. $\langle A_p, m_p \rangle \Rightarrow_p s'_p$, we have $s'_p = \langle A'_p, m_p \rangle = \langle A'_p, h(m) \rangle$, with $A'_p \in \textsc{rep}(A_p - \{\mathsf{State}(temp)\}, T)$.

On the other hand, since $A_p = h(A)$, from Lemma C.0.3 we get that for every $A'' \in \textsc{rep}(A - \{\mathsf{State}(temp)\}, T)$, $h(A'') \in \textsc{rep}(A_p - \{\mathsf{State}(temp)\}, T)$. We therefore obtain that, for every state $\langle A', m \rangle$ s.t. $\langle A, m \rangle \Rightarrow \langle A', m \rangle$, we have $\langle A_p, m_p \rangle \Rightarrow_p \langle h(A'), m_p \rangle = \langle h(A'), h(m) \rangle$.

Finally, notice that, by construction A' and A'_p do not contain $\mathsf{State}(temp)$. The claim is therefore proven by inductively applying Case 1 over A', A'_p, and h.

The other direction can be proven in the symmetric way.

Case 2 (repair step) - certain semantics: $\mathsf{State}(temp) \in A$. By hypothesis, $A_p = h(A)$, and since $h(rep)$, $\mathsf{State}(temp) \in A_p$ as well. Notice that h is syntactically applied over the ABoxes A and A_p without involving the TBox T, and therefore it can be applied also when such ABoxes are T-inconsistent. On the one hand, by construction of the transition system under the certain repair semantics, we therefore know that:

1. there exists exactly one $s' = \langle A', m \rangle$ s.t. $\langle A, m \rangle \Rightarrow s'$, where $A' = \bigcap_{A^r \in \textsc{rep}(A - \{\mathsf{State}(temp)\}, T)} A^r$;
2. there exists exactly one $s'_p = \langle A'_p, m_p \rangle = \langle A'_p, h(m) \rangle$ s.t. $\langle A_p, m_p \rangle \Rightarrow_p s'_p$, where $A'_p = \bigcap_{A^r_p \in \textsc{rep}(A_p - \{\mathsf{State}(temp)\}, T)} A^r_p$.

On the other hand, since $A_p = h(A)$, from Lemma C.0.3 we get that $A^r \in \textsc{rep}(A - \{\mathsf{State}(temp)\}, T)$ iff $h(A^r) \in \textsc{rep}(A_p - \{\mathsf{State}(temp)\}, T)$. As a consequence, $A'_p = \bigcap_{A^r \in \textsc{rep}(A - \{\mathsf{State}(temp)\}, T)} h(A^r) = h(\bigcap_{A^r \in \textsc{rep}(A - \{\mathsf{State}(temp)\}, T)} A^r) = h(A')$. Finally, notice that, by construction A' and A'_p do not contain $\mathsf{State}(temp)$. The claim is therefore proven by inductively applying Case 1 over A', A'_p, and h. □

With Lemma C.1.1 at hand, we can easily reconstruct the proof of Theorem 3.0.4 (given in Section B.1) for b- and c-KABs. Since $\mu\mathcal{L}_A^{\mathrm{IT}}$ is a fragment of $\mu\mathcal{L}_A^{\mathrm{EQL}}$, we get the result.

D. KABs under Extended Repair Semantics

D.1. Proof of Theorem 5.2.1

Given an eb-KAB (ec-KAB respectively) $\mathcal{K}$, we introduce the *pruning* $\Theta_{\mathcal{K}}$ of the transition system $\Upsilon^{eb}_{\mathcal{K}}$ ($\Upsilon^{ec}_{\mathcal{K}}$ resp.), as the transition system constructed following the extended bold (extended certain, resp.) repair semantics, but by relying on the transition relation P-EXEC$_{\mathcal{K}}$ (as defined in Section B.1) in place of EXEC$_{\mathcal{K}}$. To prove $\Theta_{\mathcal{K}} \approx \Upsilon^{eb}_{\mathcal{K}}$ ($\Theta_{\mathcal{K}} \approx \Upsilon^{ec}_{\mathcal{K}}$ resp.), one can follow step by step the line of reasoning of Section C.1, taking into consideration the fact that Viol concept assertions are inserted into the ABoxes produced by a repair step. It can be easily noticed that such assertions do not introduce any additional complication. Remember, in fact, that given an ABox A, these assertions are produced by computing the set VIOL(A, T), which is in turn produced by issuing a series of closed first-order queries over A, considered as a database of facts. Consequently, given two ABoxes A and A_p and a bijection h s.t. $A_p = h(A)$, VIOL$(A, T) =$ VIOL$(h(A), T) =$ VIOL(A_p, T).

E. Weakly Acyclic KABs

Weakly acyclic KABs are inspired by weakly acyclic tuple-generating dependencies in data exchange [15]. As in data exchange, in our setting weak acyclicity is a property defined over a *dependency graph*, constructed from the KAB's specification. In particular, the dependency graph captures the transfer of individuals from one state to the next state, differentiating between the case of copy, and the case of service calls. In fact, the latter case leads to possibly introduce fresh values into the system. The dependency graph is defined as a variation of the definitions given in [2] and [1].

Given a KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$, we define its *dependency graph* $G = \langle V, E \rangle$ as follows:

1. Nodes are defined starting from T. More specifically, we have one node $\langle N, 1 \rangle \in V$ for each concept N in T, and two nodes $\langle P, 1 \rangle, \langle P, 2 \rangle \in V$ for every role P in T (reflecting the fact that roles are binary relations, i.e., have two components).
2. Edges are defined starting from the effect specifications in Γ. We discuss the case of two concept assertions, but In particular:
   a) an ordinary edge $\langle N_1, 1 \rangle \rightarrow \langle N_2, 1 \rangle$ is contained in E if there exists an action $\gamma \in \Gamma$, an effect specification

   $$[q^+] \wedge Q^- \rightsquigarrow A'$$

   in γ, and a variable or parameter x s.t. $N_1(x)$ appears in $rew(q^+, T)$ (i.e., in the perfect rewriting of q^+ w.r.t. T), and $N_2(x)$ appears in A' (similarly for nodes corresponding to role assertions).
   b) a special edge $\langle N_1, 1 \rangle \xrightarrow{*} \langle N_2, 1 \rangle$ is contained in E if there exists an action $\gamma \in \Gamma$, an effect specification

   $$[q^+] \wedge Q^- \rightsquigarrow A'$$

   in γ, and a variable or parameter x s.t. $N_1(x)$ appears in $rew(q^+, T)$, and $N_2(f(\ldots, x, \ldots))$ appears in A' (similarly for nodes corresponding to role assertions).

A KAB $\mathcal{K}$ is *weakly acyclic* if its dependency graph has no cycle going through a special edge.

E.1. Proof of Theorem 6.0.2

To prove the theorem, we resort to the approach discussed in [2] and [1], adapting it so as to deal with inconsistency. More specifically, the main steps to prove the results are as follows:

1. Given a KAB $\mathcal{K}$, we introduce its *consistent approximant* $\mathcal{K}^p$ and positive dominant $\mathcal{K}^+$, which incrementally simplify $\mathcal{K}$ while maintaining the same dependency graph.
2. We show that when $\mathcal{K}$ is weakly acyclic, then it is run-bounded.
3. We show that $\mathcal{K}^+$ "dominates" $\mathcal{K}^p$ under all semantics discussed in the paper, i.e., the active domain of the transition system for $\mathcal{K}$ is always contained in the active domain of the transition system for $\mathcal{K}^+$.
4. We do the same for $\mathcal{K}$ w.r.t. $\mathcal{K}^p$, thus transferring the weak acyclicity property from $\mathcal{K}^+$ to $\mathcal{K}$.

Technically, given a KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$, we define the *consistent approximant* $\mathcal{K}^p$ of $\mathcal{K}$ as a KAB $= (T_p, A_0^p, \Gamma^p, \Pi)$, where A_0^p and Γ^p are obtained as follows:

- $A_0^p = A_0 \cup \{\mathsf{Viol}(d) \mid \exists t \in T_n \cup T_f \text{ s.t. } d = \textsc{label}(t)\}$; i.e., A_0^p saturates A_0 with all possible violations of negative inclusion and functionality assertions in T.
- For every action $\alpha(p_1, \ldots, p_n) : \{e_1, \ldots, e_m\} \in \Gamma$ we have $\alpha(p_1, \ldots, p_n) : \{e_v, e_1, \ldots, e_m\} \in \Gamma^p$, where $e_v = \mathsf{Viol}(x) \rightsquigarrow \{\mathsf{Viol}(x)\}$ copies all Viol assertions.

Notice that the TBox of the consistent approximant is constituted by the positive inclusion assertions of the original TBox.

Starting from the consistent approximant, we define the *positive dominant* $\mathcal{K}^+$ of $\mathcal{K}$ as a KAB $= (T_p, A_0^p, \Gamma^+, \Pi^+)$, where Γ^+ and Π^+ are obtained as follows:

- For each action $\alpha(p_1, \ldots, p_n) : \{e_1, \ldots, e_m\} \in \Gamma^p$ we have $\alpha^+() : \{e_1^+, \ldots, e_m^+\} \in \Gamma^+$ where, given $e_i = [q_i^+] \wedge Q^- \rightsquigarrow A_i'$, we have $e_i^+ = [q_i^+] \rightsquigarrow A_i'$.
- For each condition-action rule $Q \mapsto \alpha(p_1, \ldots, p_n) \in \Pi$, we have $\mathit{true} \mapsto \alpha^+() \in \Pi^+$.

It is easy to show that the dependency graphs of $\mathcal{K}$, $\mathcal{K}^p$ and $\mathcal{K}^+$ coincide, and therefore $\mathcal{K}$ is weakly acyclic iff $\mathcal{K}^p$ is weakly acyclic iff $\mathcal{K}^+$ is weakly acyclic.

Theorem E.1.1. *Given KAB $\mathcal{K}$, if $\mathcal{K}$ is weakly acyclic then its positive dominant $\mathcal{K}^+$ is run-bounded.*

Proof. By compiling away the TBox of $\mathcal{K}^+$ exploiting the first-order rewritability of *DL-Lite*$_\mathcal{A}$, the obtained KAB exactly corresponds to the notion of *positive approximant* defined for relational Data-Centric Dynamic Systems in [2]. The proof is then directly obtained from the proof of Theorem 4.7 in [2]. □

To show that Theorem E.1.1 extends to the KAB itself under each of the semantics considered in this paper, we first introduce the notion of *dominance* between transition systems. Technically, a transition system Υ_1 is *dominated by* Υ_2 if, for every run τ_1 in Υ_1 there exists a run τ_2 in Υ_2 s.t. for all pairs of states $\tau_1(i)$ and $\tau_2(i)$, we have $\mathit{abox}(\tau_1(i)) \subseteq \mathit{abox}(\tau_2(i))$. By definition, we consequently have that if Υ_2 is run-bounded, then Υ_1 is run-bounded as well. This shows that, to prove run-boundedness of a transition system, it is sufficient to prove that such a transition system is dominated by a run-bounded transition system.

With this machinery at hand, we are now able to prove the following two key lemmas, which respectively show that for any semantics considered in this paper, the consistent approximant is dominated by the positive dominant, and dominates the original KAB.

Lemma E.1.2. *For any KAB $\mathcal{K}$, we have that:*

1. $\Upsilon^{s}_{\mathcal{K}^p}$ *is dominated by* $\Upsilon^{s}_{\mathcal{K}^+}$;
2. $\Upsilon^{b}_{\mathcal{K}^p}$ *is dominated by* $\Upsilon^{b}_{\mathcal{K}^+}$;
3. $\Upsilon^{c}_{\mathcal{K}^p}$ *is dominated by* $\Upsilon^{c}_{\mathcal{K}^+}$;
4. $\Upsilon^{eb}_{\mathcal{K}^p}$ *is dominated by* $\Upsilon^{eb}_{\mathcal{K}^+}$;
5. $\Upsilon^{ec}_{\mathcal{K}^p}$ *is dominated by* $\Upsilon^{ec}_{\mathcal{K}^+}$.

Proof. We discuss claim 1 and claims 2-5 separately.

Each claim can be obtained by proving the following stronger claim: for every run τ in $\Upsilon^{s}_{\mathcal{K}^p}$ (resp., $\Upsilon^{b}_{\mathcal{K}^p}$, $\Upsilon^{c}_{\mathcal{K}^p}$, $\Upsilon^{eb}_{\mathcal{K}^p}$, $\Upsilon^{ec}_{\mathcal{K}^p}$), there exists a run τ^+ in $\Upsilon^{s}_{\mathcal{K}^+}$ (resp., $\Upsilon^{b}_{\mathcal{K}^+}$, $\Upsilon^{c}_{\mathcal{K}^+}$, $\Upsilon^{eb}_{\mathcal{K}^+}$, $\Upsilon^{ec}_{\mathcal{K}^+}$) s.t. for all pairs of state $\tau(i) = \langle A_i, m_i \rangle$ and $\tau^+(i) = \langle A^+_i, m^+_i \rangle$, we have:

1. $A_i \subseteq A^+_i$;
2. m^+_i extends m_i;
3. for the mappings mentioned in m^+_i but not in m_i, m^+_i "agrees" with the maps contained in the suffix of $\tau(i)$, i.e.,

$$m^+_i|_{C_i} = (\bigcup_{j>i} m_j)|_{C_i}$$

where $C_i = \textsc{dom}(m^+_i) \cap \bigcup_{j>i} \textsc{dom}(m_j)$.

Claim 1. Thanks to the first-order rewritability of *DL-Lite*$_\mathcal{A}$, $\mathcal{K}^p$ and $\mathcal{K}^+$ can be correspondingly represented as a Data-Centric Dynamic System in the sense of [2]. The proof is then directly obtained from the proof of Lemma 4.1 in [2].

Claim 2-5. The claims can be easily shown by observing that $\mathcal{K}^p$ and $\mathcal{K}^+$ never produce an ABox that is T_p-inconsistent, since they only consider positive inclusion assertions. Consequently, under each of the repair semantics, the repair service does not affect the current ABox: it simply generates a unique successor that contains the same ABox and service call map produced by the previous action step. This shows that $\Upsilon^{b}_{\mathcal{K}^p} = \Upsilon^{c}_{\mathcal{K}^p} = \Upsilon^{eb}_{\mathcal{K}^p} = \Upsilon^{ec}_{\mathcal{K}^p}$ and that $\Upsilon^{b}_{\mathcal{K}^+} = \Upsilon^{c}_{\mathcal{K}^+} = \Upsilon^{eb}_{\mathcal{K}^+} = \Upsilon^{ec}_{\mathcal{K}^+}$. To get the claims, given the current state $\langle A, m \rangle$ in $\Upsilon^{b}_{\mathcal{K}^p}$, we specifically discuss the case in which $\mathsf{State}(\mathit{temp}) \notin A$, and the case in which $\mathsf{State}(\mathit{temp}) \in A$:

(base case) Trivial, because the initial states of $\Upsilon^{b}_{\mathcal{K}^p}$ and $\Upsilon^{b}_{\mathcal{K}^+}$ coincide (they are both equal to $\langle A^p_0, \emptyset \rangle$).

(case 1 - action step) Since it cannot be the case that the state produced after an action step is T_p-inconsistent, then the proof exactly follows the one for Claim 1.

(case 2 - repair step) Consider $\tau(i) = \langle A, m \rangle$ and $\tau^+(i) = \langle A^+, m^+ \rangle$ s.t.: 1. $\mathsf{State}(\mathit{temp}) \in A$ and $\mathsf{State}(\mathit{temp}) \in A^+$; 2. A and A^+ satisfy condition 1; 3. m and m^+ satisfy conditions 2 and 3. Since A and A^+ are T_p-consistent, then there is a unique successor $\langle A - \{\mathsf{State}(\mathit{temp})\}, m \rangle$ of $\tau(i)$ in $\Upsilon^{b}_{\mathcal{K}^p}$, and a unique successor $\langle A^+ - \{\mathsf{State}(\mathit{temp})\}, m^+ \rangle$ of $\tau^+(i)$ in $\Upsilon^{b}_{\mathcal{K}^+}$. It is trivial to see that these successors satisfy the three conditions of the claim above.

□

Lemma E.1.3. *For any KAB $\mathcal{K}$, we have that:*

1. $\Upsilon^s_{\mathcal{K}}$ *is dominated by* $\Upsilon^s_{\mathcal{K}^p}$;
2. $\Upsilon^b_{\mathcal{K}}$ *is dominated by* $\Upsilon^b_{\mathcal{K}^p}$;
3. $\Upsilon^c_{\mathcal{K}}$ *is dominated by* $\Upsilon^c_{\mathcal{K}^p}$;
4. $\Upsilon^{eb}_{\mathcal{K}}$ *is dominated by* $\Upsilon^{eb}_{\mathcal{K}^p}$;
5. $\Upsilon^{ec}_{\mathcal{K}}$ *is dominated by* $\Upsilon^{ec}_{\mathcal{K}^p}$.

Proof. We discuss each claim separately, by referring to the three inductive conditions defined in the stronger claim of the proof of Lemma E.1.2.

Case 1. Trivial, because $\Upsilon^s_{\mathcal{K}}$ is a fragment of $\Upsilon^s_{\mathcal{K}^p}$: it does not contain the portions of $\Upsilon^s_{\mathcal{K}^p}$ that are generated starting from a T-inconsistent (but always T_p-consistent) ABox.

Case 2. The base case is trivial, because the initial state of $\Upsilon^b_{\mathcal{K}}$ is $\langle A_0, \emptyset \rangle$, the initial state of $\Upsilon^b_{\mathcal{K}^p}$ is $\langle A^p_0, \emptyset \rangle$, and by construction $A_0 \subseteq A^p_0$.

The inductive case for an action step can be proven exactly in the same way discussed in the proof of Lemma E.1.2 - Claim 1.

We then focus on the inductive case for a repair step. Consider $\tau(i) = \langle A, m \rangle$ in $\Upsilon^b_{\mathcal{K}}$ and $\tau^p(i) = \langle A^p, m^p \rangle$ in $\Upsilon^b_{\mathcal{K}^p}$, s.t. conditions 1, 2 and 3 hold. By construction, we know that:

- every successor of $\langle A, m \rangle$ in $\Upsilon^b_{\mathcal{K}}$ has the form $\langle A', m \rangle$, where $A' \in \textsc{rep}(A - \mathsf{State}(\mathit{temp}), T)$;
- $\langle A^p, m^p \rangle$ has a unique successor $\langle A^p - \{\mathsf{State}(\mathit{temp})\}, m^p \rangle$ in $\Upsilon^b_{\mathcal{K}^p}$.

Since the service call maps do not change, the successors continue to obey to conditions 2 and 3. Furthermore, by definition of $\textsc{rep}()$, we know that $A' \subseteq A$ and, by hypothesis, that $A \subseteq A^p$. Consequently, $A' \subseteq A_p$, and therefore also condition 1 is satisfied.

Case 3. The base case and the inductive case for an action step are as in Case 2. We then focus on the inductive case for a repair step. Consider $\tau(i) = \langle A, m \rangle$ in $\Upsilon^b_{\mathcal{K}}$ and $\tau^p(i) = \langle A^p, m^p \rangle$ in $\Upsilon^b_{\mathcal{K}^p}$, s.t. conditions 1, 2 and 3 hold. By construction, we know that:

- $\langle A, m \rangle$ has a unique successor $\langle A', m \rangle$ in $\Upsilon^b_{\mathcal{K}}$, where $A' = \bigcap_{A^r \in \textsc{rep}(A - \{\mathsf{State}(\mathit{temp})\}, T)} A^r$;
- $\langle A^p, m^p \rangle$ has a unique successor $\langle A^p - \{\mathsf{State}(\mathit{temp})\}, m^p \rangle$ in $\Upsilon^b_{\mathcal{K}^p}$.

Since the service call maps do not change, the successors continue to obey to conditions 2 and 3. Furthermore, by definition we have $A' \subseteq A$ and, by hypothesis, we know that $A \subseteq A^p$. Consequently, $A' \subseteq A_p$, and therefore also condition 1 is satisfied.

Case 4. This case is directly obtained from Case 2, and from the observation that, by construction, each ABox of the consistent approximant contains all the possible Viol assertions, since they are asserted in the initial state, and copied by means of a specific effect contained in each of its actions. Therefore, after a repair step, it is guaranteed that the ABox obtained in $\Upsilon^{eb}_{\mathcal{K}}$ is a subset of the corresponding ABox in $\Upsilon^{eb}_{\mathcal{K}^p}$.

Case 5. This case is directly obtained from Case 3 and the observation done for Case 4. □

The proof of Theorem 4.2.1 is finally obtained by combining Theorem E.1.1 and the composition of Lemma E.1.3 with Lemma E.1.2, thanks to transitivity of domination.

F. KABs with Consistent Query Answering

As mentioned in the conclusion of the paper, an orthogonal approach to manage inconsistency would be to make the KAB itself inconsistency-tolerant. More specifically, we can conceive a KAB that admits inconsistent ABoxes, and that replaces the standard query answering service with an inconsistency-tolerant querying service, able to extract meaningful answers even in presence of inconsistent information.

In the following, we rely for this purpose on the standard notion of *consistent query answering* in databases [3], which has been extended to the knowledge base setting in [18]. More specifically, we introduce the following query answering service, which corresponds to the notion of AR-consistent entailment in [18] (Definition 3).

Given an UCQ q, the *consistent-query answer* to q over (T, A) is the set $cqa(q, T, A)$ of substitutions σ of the free variables of q with constants in $\textsc{adom}(A)$ s.t., for every repair $A_r \in \textsc{rep}(A, T)$, $q\sigma$ evaluates to true in every model of (T, A_r). Observe that, when A is T-consistent, the consistent-query answers coincide with the certain answers.

Like for certain answers, we extend the notion of consistent-query answer to ECQ as follows: given an ECQ Q, the *consistent-query answer to Q over* (T, A), is the set $\textsc{cqa}(Q, T, A)$ of tuples of constants in $\textsc{adom}(A)$ defined by composing the consistent-query answers $cqa(q, T, A)$ of UCQs q through first-order constructs, and interpreting existential variables as ranging over $\textsc{adom}(A)$.

F.1. Inconsistency-tolerant KABs

We introduce the *inconsistency-tolerant* semantics for KABs as the variation of the standard semantics where:

- all queries are answered using consistent-query answering instead of certain answers (i.e., by replacing every $\textsc{ans}(Q, T, A)$ with $\textsc{cqa}(Q, T, A)$);
- an action with parameters is applied even if the resulting ABox is T-inconsistent (in fact, consistent-query answering makes it possible to query such an inconsistent ABox in a meaningful way).

We call *it-KAB* a KAB interpreted under the inconsistency-tolerant semantics. Given an it-KAB $\mathcal{K}$, we denote with $\Upsilon^{it}_{\mathcal{K}}$ the transition system describing its execution semantics.

In order to specify temporal/dynamic properties over it-KABs, also the $\mu\mathcal{L}^{\text{EQL}}_A$ logic must be adapted, making it able to query even T-inconsistent ABoxes in a meaningful way. In particular, we introduce the logic $\mu\mathcal{L}^{\text{CQA}}_A$ that is syntactically equivalent to $\mu\mathcal{L}^{\text{EQL}}_A$, but redefines the semantics of local EQL queries Q as follows:

$$(Q)^{\Upsilon}_{v,V} = \{s \in \Sigma \mid \textsc{cqa}(Qv, T, abox(s)) = \mathsf{true}\}$$

F.2. Verification of Inconsistency-Tolerant KABs

In this Section, we show that the decidability results presented for the repair semantics seamlessly apply to it-KABs as well.

Lemma F.2.1. *Consider two knowledge bases (T, A_1) and (T, A_2), s.t. there exists a bijection h with $A_2 = h(A_1)$. For every EQL query q, we have $\langle d_1, \ldots, d_n \rangle \in \text{CQA}(q, T, A_1)$ iff $\langle h(d_1), \ldots, h(d_n) \rangle \in \text{CQA}(h(q), T, h(A_1))$.*

Proof. This result is a direct consequence of the combination of Lemmas A.0.3 and C.0.3. □

Theorem F.2.2. *Verification of $\mu\mathcal{L}_A^{\text{CQA}}$ properties over run-bounded it-KABs is decidable.*

Proof. By inspecting the proofs of Theorem 3.0.4 (given in Appendix B.1), we observe that the possibility of constructing a faithful finite-state abstraction for a run-bounded KAB depends on the fact that its execution semantics produce bisimilar runs starting from isomorphic states. This key property, in turn, relies on the fact that the query answering service does not distinguish between isomorphic states. Since this holds for consistent-query answers as well (see Lemma F.2.1), we can follow, step-by-step, the same proof given in Appendix B.1. □

Theorem F.2.3. *Given a weakly acyclic KAB $\mathcal{K}$, we have that $\Upsilon^{it}_{\mathcal{K}}$ is run-bounded.*

Proof. Consider the consistent approximant $\mathcal{K}^p$ of $\mathcal{K}$. From Lemma E.1.2, we know that $\Upsilon^s_{\mathcal{K}^p}$ is dominated by $\Upsilon^s_{\mathcal{K}^+}$. By inspecting the proof of this claim, which in turn refers to the proof of Lemma 4.1 in [2], we know that this is the case because, state by state, the answers extracted by $\mathcal{K}^p$ are contained in the ones extracted by $\mathcal{K}^+$.

We now observe that, by definition, given a TBox T, an ABox A and an EQL query Q, $\text{CQA}(Q, T, A) \subseteq \text{CQA}(Q, T_p, A) = \text{ANS}(Q, T_p, A)$. The equality $\text{CQA}(Q, T_p, A) = \text{ANS}(Q, T_p, A)$ holds because every ABox is consistent with T_p, and the only repair of a consistent ABox is the ABox itself.

Consequently, we can apply the same line of reasoning used in the proof of Lemma 4.1 in [2], showing that $\Upsilon^{it}_{\mathcal{K}}$ is dominated by $\Upsilon^s_{\mathcal{K}^p}$. By applying Lemma E.1.2 and transitivity of domination, this in turn implies that $\Upsilon^{it}_{\mathcal{K}}$ is dominated by $\Upsilon^s_{\mathcal{K}^+}$. By recalling Theorem E.1.1 we finally get the result. □