



HAL
open science

Verification of Inconsistency-Aware Knowledge and Action Bases

Diego Calvanese, Evgeny Kharlamov, Marco Montali, Ario Santoso, Dmitriy
Zheleznyakov

► **To cite this version:**

Diego Calvanese, Evgeny Kharlamov, Marco Montali, Ario Santoso, Dmitriy Zheleznyakov. Verification of Inconsistency-Aware Knowledge and Action Bases. IJCAI, 2013, Beijing, China. hal-00817495

HAL Id: hal-00817495

<https://inria.hal.science/hal-00817495>

Submitted on 24 Apr 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Verification of Inconsistency-Aware Knowledge and Action Bases*

Diego Calvanese, Evgeny Kharlamov, Marco Montali, Ario Santoso, Dmitriy Zheleznyakov

KRDB Research Centre for Knowledge and Data

Free University of Bozen-Bolzano

lastname@inf.unibz.it

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.

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 [Deutsch *et al.*, 2009; Vianu, 2009; Bagheri Hariri *et al.*, 2012; Calvanese *et al.*, 2012; Limonad *et al.*, 2012]. 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 [Clarke *et al.*, 1999] nor most of the current techniques for infinite-state model checking [Burkart *et al.*, 2001] apply to this case.

Along this line, *Knowledge and Action Bases (KABs)* [Bagheri Hariri *et al.*, 2012] 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.

*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.

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., [Deutsch *et al.*, 2007; Deutsch *et al.*, 2009; Bagheri Hariri *et al.*, 2013]). 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 [Winslett, 1990; Eiter and Gottlob, 1992; Flouris *et al.*, 2008; Calvanese *et al.*, 2010], and, in particular, on the notion of knowledge base repair [Lembo *et al.*, 2010], 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 an extended version of this paper [Calvanese *et al.*, 2013].

2 Preliminaries

2.1 *DL-Lite_A* Knowledge Bases

For expressing knowledge bases, we use *DL-Lite_A* [Poggi *et al.*, 2008; Calvanese *et al.*, 2009]. The syntax of *concept* and *role expressions* in *DL-Lite_A* is as follows

$$B \rightarrow N \mid \exists R \quad R \rightarrow P \mid P^-$$

where N denotes a *concept name*, P a *role name*, and P^- an *inverse role*. A *DL-Lite_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 $(\text{funct } R)$.

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 $\bigvee_{1 \leq i \leq n} \exists \vec{y}_i. \text{conj}_i(\vec{x}, \vec{y}_i)$ with free variables \vec{x} and existentially quantified variables $\vec{y}_1, \dots, \vec{y}_n$. Each $\text{conj}_i(\vec{x}, \vec{y}_i)$ 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 $i \in \{1, \dots, n\}$. The *(certain) answers to q over (T, A)* is the set $\text{ans}(q, T, A)$ 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)* [Calvanese *et al.*, 2007a], 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 $\text{ans}(q, T, A)$ of UCQs q through first-order constructs, and interpreting existential variables as ranging over $\text{ADOM}(A)$.

Finally, we recall that *DL-Lite_A* enjoys the *FO rewritability* property, which states that for every UCQ q , $\text{ans}(q, T, A) = \text{ans}(\text{rew}(q), \emptyset, A)$, where $\text{rew}(q)$ is a UCQ computed by the reformulation algorithm in [Calvanese *et al.*, 2009]. Notice that this algorithm can be extended to ECQs [Calvanese *et al.*, 2007a], 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 [Bagheri Hariri *et al.*, 2012]. 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_A TBox T* and an initial *T-consistent DL-Lite_A ABox A₀*. 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* $\gamma \in \Gamma$ generates a new ABox A' from the current ABox A , by adding or deleting assertions. 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 $\{e_1, \dots, e_n\}$ 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 [Calvanese *et al.*, 2013]).

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 2.1. $\mathcal{K} = (T, A_0, \Gamma, \Pi)$ 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) $\Pi = \{\text{true} \mapsto \gamma_1, C(y) \mapsto \gamma_2(y)\}$. \square

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 $(\mathcal{C}, T, \Sigma, s_0, \text{abox}, \Rightarrow)$, 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 [Bagheri Hariri *et al.*, 2012], 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 [Bagheri Hariri *et al.*, 2013] 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 $\mathbb{S}\mathcal{C} = \{f(v_1, \dots, v_n) \mid f/n \in \mathcal{F} \text{ and } \{v_1, \dots, v_n\} \subseteq \mathcal{C}\}$, and define a *service call map* as a partial function $m : \mathbb{S}\mathcal{C} \rightarrow \mathcal{C}$.

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 $\gamma(p_1, \dots, p_r) : \{e_1, \dots, e_k\}$ be an action in Γ with parameters p_1, \dots, p_r , and $e_i = [q_i^+] \wedge Q_i^- \rightsquigarrow E_i$. Let σ be a substitution for p_1, \dots, 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 $\langle p_1, \dots, p_r \rangle \sigma \in \text{ANS}(Q, A)$. We denote with $\text{DO}(T, A, \gamma\sigma)$ the set of facts obtained by evaluating the effects of action γ with parameters σ on ABox A , so as to *progress* (cf. planning [Ghallab *et al.*, 2004]) the system from the current state to the next:

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

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 $\text{DO}()$ 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 $\text{CALLS}(E)$ the set of calls it contains, and with $\text{EVALS}(T, A, \gamma\sigma)$ the set of substitutions that replace all service calls in $\text{DO}(T, A, \gamma\sigma)$ with values in \mathcal{C} :

$$\text{EVALS}(T, A, \gamma\sigma) = \{\theta \mid \theta \text{ is a total function} \\ \theta : \text{CALLS}(\text{DO}(T, A, \gamma\sigma)) \rightarrow \mathcal{C}\}.$$

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

Example 3.1. Consider our running example (Example 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 . \square

Given a KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$, we employ $\text{DO}()$ and $\text{EVALS}()$ to define a transition relation $\text{EXEC}_{\mathcal{K}}$ connecting two states through the application of an action with parameter assignment. In particular, given an action with parameter assignment $\gamma\sigma$, we have $\langle \langle A, m \rangle, \gamma\sigma, \langle A', m' \rangle \rangle \in \text{EXEC}_{\mathcal{K}}$ 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 $\theta \in$

$\text{EVALS}(T, A, \gamma\sigma)$ such that θ and m agree on the common values in their domains (so as to realize the deterministic service semantics); (iii) $A' = \text{DO}(T, A, \gamma\sigma)\theta$; (iv) $m' = m \cup \theta$ (i.e., the history of issued service calls is updated).

Standard transition system. The *standard transition system* $\Upsilon_{\mathcal{K}}^{\mathbb{S}}$ for KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$ is a (possibly infinite-state) transition system $(\mathcal{C}, T, \Sigma, s_0, \text{abox}, \Rightarrow)$ where: (i) $s_0 = \langle A_0, \emptyset \rangle$; (ii) $\text{abox}(\langle A, m \rangle) = A$; (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 $\langle \langle A, m \rangle, \gamma\sigma, \langle A', m' \rangle \rangle \in \text{EXEC}_{\mathcal{K}}$, 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 3.2. Consider \mathcal{K} of Example 2.1 and its standard transition system $\Upsilon_{\mathcal{K}}^{\mathbb{S}}$. As discussed in Example 3.1, in state $s_0 = \langle A_0, \emptyset \rangle$ only γ_2 is applicable with parameter a . Since $\text{DO}(T, A_0, \gamma_2(a)) = \{G(f(a))\}$, $\Upsilon_{\mathcal{K}}^{\mathbb{S}}$ 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} . \square

Verification Formalism. To specify sophisticated temporal properties over KABs, we resort to the first-order variant of μ -calculus [Stirling, 2001; Park, 1976] defined in [Bagheri Hariri *et al.*, 2012]. This variant, here called $\mu\mathcal{L}_A^{\text{EQL}}$, 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:

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

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: $\forall x.\Phi = \neg(\exists x.\neg\Phi)$, $\Phi_1 \vee \Phi_2 = \neg(\neg\Phi_1 \wedge \neg\Phi_2)$, $\langle \neg \rangle \Phi = \neg(\neg)\neg\Phi$, 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 $(\mathcal{C}, T, \Sigma, s_0, \text{abox}, \Rightarrow)$. 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 [Bagheri Hariri *et al.*, 2012]:

$$(Q)_{v,V}^{\Upsilon} = \{s \in \Sigma \mid \text{ANS}(Qv, T, \text{abox}(s)) = \text{true}\} \\ (\exists x.\Phi)_{v,V}^{\Upsilon} = \{s \in \Sigma \mid \exists d.d \in \text{ADOM}(\text{abox}(s)) \\ \text{and } s \in (\Phi)_{v[x/d],V}^{\Upsilon}\}$$

Here, Qv stands for the query obtained from Q by substituting its free variables according to v . When Φ is a closed formula, $(\Phi)_{v,V}^{\Upsilon}$ 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 [Bagheri Hariri *et al.*, 2012] to the case of S-KABs, obtaining that verification is undecidable even for the very simple temporal reachability property $\mu Z.(\mathbf{N}(a) \vee (\neg)Z)$, with \mathbf{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* [Bagheri Hariri *et al.*, 2013]. Given an S-KAB \mathcal{K} , a run $\tau = s_0 s_1 \dots$ of $\Upsilon_{\mathcal{K}}^{\text{S}}$ is *bounded* if there exists a finite bound b s.t. $|\bigcup_{s \text{ state of } \tau} \text{ADOM}(abox(s))| < b$. 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 3.3. *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 [Clarke *et al.*, 1999; Stirling, 2001], 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 [Winslett, 1990; Eiter and Gottlob, 1992; Flouris *et al.*, 2008; Calvanese *et al.*, 2010], and, in particular, on the notion of *ABox repair*, cf. [Bertossi, 2006; Lembo *et al.*, 2010].

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 [Eiter and Gottlob, 1992] and are applicable to the context of DL ontologies [Lembo *et al.*, 2010].

- 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 $\text{REP}(A, T)$ 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: $A' = \bigcap_{A'' \in \text{REP}(A, T)} A''$. 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 4.1. *Continuing Example 3.1, consider the T -inconsistent state $\langle A', \emptyset \rangle$ obtained from $\gamma_1()$ in A_0 . Its two b-repairs are $\text{REP}(A', T) = \{A_1, A_2\}$ with $A_1 = \{C(a)\}$, $A_2 = \{D(a)\}$. Its c-repair is $\bigcap_{A \in \text{REP}(A', T)} A = \{C(a)\} \cap \{D(a)\} = \emptyset$. \square*

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 $\text{State}(temp)$, which is an ABox statement with a fresh concept name State and a fresh constant $temp$, s.t.: if $\text{State}(temp)$ 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* $\Upsilon_{\mathcal{K}}^b$ (resp. *c-transition system* $\Upsilon_{\mathcal{K}}^c$) for a KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$ is a (possibly infinite-state) transition system $(\mathcal{C}, T, \Sigma, s_0, abox, \Rightarrow)$ 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 $\text{State}(temp) \notin A$, 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. $\langle \langle A, m \rangle, \gamma\sigma, \langle A', m' \rangle \rangle \in \text{EXEC}_{\mathcal{K}}$, let $A'' = A' \cup \{\text{State}(temp)\}$, 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 $\text{State}(temp) \in A$, then for b-repair A' (resp. c-repair A') of $A - \{\text{State}(temp)\}$ 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 4.2. 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 $A'_r = \{A' \cup \{\text{State}(\text{temp})\}\}$. 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 3.2), we have $\langle A_0, \emptyset \rangle \Rightarrow \langle A'' \cup \{\text{State}(\text{temp})\}, m \rangle \Rightarrow \langle A'', m \rangle$, reflecting that in this case the repair service just maintains the resulting ABox unaltered. \square

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 4.2). 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 4.2 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 $\mu\mathcal{L}_A^{\text{IT}}$, which is a fragment of $\mu\mathcal{L}_A^{\text{EQL}}$ defined as:

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

Beside the standard abbreviations introduced for $\mu\mathcal{L}_A^{\text{EQL}}$, we also make use of the following: $\langle \rightarrow \rangle \Phi = \neg[\rightarrow] \neg\Phi$, and $[\rightarrow] \Phi = \neg\langle \rightarrow \rangle \neg\Phi$. This logic can be used to express interesting properties over b- and c-KABs, exploiting different combinations of the $\langle \rightarrow \rangle$ and $[\rightarrow]$ 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, $\mu Z.(\Phi \vee \langle \rightarrow \rangle Z)$ models the “optimistic” reachability of Φ , stating that there exists a sequence of action and repair steps, s.t. Φ eventually holds. Conversely, $\mu Z.(\Phi \vee \langle \rightarrow \rangle [\rightarrow] Z)$ 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., $\nu Z.(\forall x.\text{Stud}(x) \rightarrow \mu Y.(\text{Grad}(x) \vee \langle \rightarrow \rangle Y)) \wedge [\rightarrow] Z$ 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 4.3. Verification of $\mu\mathcal{L}_A^{\text{IT}}$ 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 $DL\text{-Lite}_A$ does not allow, in general, to uniquely extract from a T -inconsistent ABox a set of individuals that are responsible for the inconsistency [Calvanese *et al.*, 2007b], its *separability* property [Calvanese *et al.*, 2007b] 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 $DL\text{-Lite}_A$ 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 $DL\text{-Lite}_A$ [Calvanese *et al.*, 2007b], 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 [Calvanese *et al.*, 2007b], given a functionality assertion (funct R), we construct the query $q_{\text{unsat}}^f(\text{funct } R) = \exists x, x_1, x_2. \eta(R, x, x_1) \wedge \eta(R, x, x_2) \wedge x_1 \neq x_2$, 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 $q_{\text{unsat}}^n(B_1 \sqsubseteq \neg B_2, T_p) = \text{rew}(T_p, \exists x. \gamma(B_1, x) \wedge \gamma(B_2, x))$, 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 $q_{\text{unsat}}^n(R_1 \sqsubseteq \neg R_2, T_p) = \text{rew}(T_p, \exists x_1, x_2. \eta(R_1, x_1, x_2) \wedge \eta(R_2, x_1, x_2))$.

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 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 $\text{VIOL}(A, T)$ of constants labeling TBox assertions in T violated by A , as:

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

Example 5.1. Consider \mathcal{K} in Example 2.1, with $T = \{C \sqsubseteq \neg D\}$, and $A' = \{D(a), C(a)\}$ in Example 3.1. Assume that $\text{LABEL}(C \sqsubseteq \neg D) = \ell$. We have $\phi = q_{\text{unsat}}^n(C \sqsubseteq \neg D, \emptyset) = \exists x. C(x) \wedge D(x)$. Since $A' \models \phi$, we get $\text{VIOL}(A', T) = \{\ell\}$.

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* $\Upsilon_{\mathcal{K}}^{\text{eb}}$ (resp. *ec-transition system* $\Upsilon_{\mathcal{K}}^{\text{ec}}$) for KAB $\mathcal{K} = (T, A_0, \Gamma, \Pi)$ as a (possibly infinite-state) transition system $(\mathcal{C}, T, \Sigma, s_0, \text{abox}, \Rightarrow)$

constructed starting from $\Upsilon_{\mathcal{K}}^b$ (resp. $\Upsilon_{\mathcal{K}}^c$) by refining the repair step as follows: if $\langle A, m \rangle \in \Sigma$ and $\text{State}(temp) \in A$, then for b-repair A' (resp. c-repair A') of $A - \{\text{State}(temp)\}$ with T , we have $\langle A'_v, m \rangle \in \Sigma$ and $\langle A, m \rangle \Rightarrow \langle A'_v, m \rangle$, where $A'_v = A' \cup \{\text{Viol}(d) \mid d \in \text{VIOL}(A', T)\}$.

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 $\mu\mathcal{L}_A^{\text{IT}}$ 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:

- $\nu Z.(\neg\exists l.\text{Viol}(l)) \wedge [\neg][\neg]Z$ says that no state of the system is manipulated by the repair service;
- $\nu Z.(\forall l.\text{Viol}(l) \rightarrow (\mu Y.\nu W.\neg\text{Viol}(l) \wedge [\neg][\neg]W \vee \langle \neg \rangle[\neg]Y) \wedge [\neg][\neg]Z)$ 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 4.3 can be seamlessly carried over to these extended repair semantics.

Theorem 5.2. *Verification of $\mu\mathcal{L}_A^{\text{IT}}$ 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 $\mu\mathcal{L}_A^{\text{EQL}}$ 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 $\Upsilon_{\mathcal{K}}^s$ if that property holds in the portion of the $\Upsilon_{\mathcal{K}}^{eb}$ (or $\Upsilon_{\mathcal{K}}^{ec}$) where no TBox assertion is violated. The absence of violation can be checked over T -consistent states by issuing the EQL query $\neg\exists x.[\text{Viol}(x)]$. 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 $\langle \neg \rangle\Psi$ with $\langle \neg \rangle\langle \neg \rangle((\neg\exists x.\text{Viol}(x)) \wedge \tau(\Psi))$, and each occurrence of $[\neg]\Psi$ with $[\neg]\langle \neg \rangle((\neg\exists x.\text{Viol}(x)) \rightarrow \tau(\Psi))$. Observe that, in τ , the choice of $\langle \neg \rangle$ for the nested operator can be substituted by $[\neg]$, 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. *Given a KAB \mathcal{K} and a $\mu\mathcal{L}_A^{\text{EQL}}$ property Φ , $\Upsilon_{\mathcal{K}}^s \models \Phi$ iff $\Upsilon_{\mathcal{K}}^{eb} \models \tau(\Phi)$ iff $\Upsilon_{\mathcal{K}}^{ec} \models \tau(\Phi)$.*

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 [Bagheri Hariri *et al.*, 2013], *run boundedness* is a semantic condition that is undecidable to check. In [Bagheri Hariri *et al.*, 2013], a sufficient, syntactic condition borrowed from *weak acyclicity* in data exchange [Fagin *et al.*, 2005] 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 [Fagin *et al.*, 2005], 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 6.1. *Given a weakly acyclic KAB \mathcal{K} , we have that $\Upsilon_{\mathcal{K}}^s, \Upsilon_{\mathcal{K}}^b, \Upsilon_{\mathcal{K}}^c, \Upsilon_{\mathcal{K}}^{eb}, \Upsilon_{\mathcal{K}}^{ec}$ are all run-bounded.*

Theorem 6.1 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 [Bertossi, 2006; Lembo *et al.*, 2010]. 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.

References

- [Bagheri Hariri *et al.*, 2012] 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.
- [Bagheri Hariri *et al.*, 2013] 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.
- [Bertossi, 2006] Leopoldo E. Bertossi. Consistent query answering in databases. *SIGMOD Record*, 35(2):68–76, 2006.
- [Burkart *et al.*, 2001] O. Burkart, D. Cauca, F. Moller, and B. Steffen. Verification of infinite structures. In *Handbook of Process Algebra*. Elsevier Science, 2001.
- [Calvanese *et al.*, 2007a] 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.
- [Calvanese *et al.*, 2007b] 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.
- [Calvanese *et al.*, 2009] 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 Information Systems – 5th Int. Summer School Tutorial Lectures (RW 2009)*, volume 5689 of *Lecture Notes in Computer Science*, pages 255–356. Springer, 2009.
- [Calvanese *et al.*, 2010] 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.
- [Calvanese *et al.*, 2012] 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.
- [Calvanese *et al.*, 2013] Diego Calvanese, Evgeny Kharlamov, Marco Montali, Ario Santoso, and Dmitriy Zheleznyakov. Verification of inconsistency-aware knowledge and action bases (extended version). KRDB Research Centre Technical Report KRDB13-2, Free University of Bozen-Bolzano, 2013. Available at <http://www.inf.unibz.it/krdp/pub/TR/KRDB13-2.pdf>.
- [Clarke *et al.*, 1999] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model checking*. The MIT Press, Cambridge, MA, USA, 1999.
- [Deutsch *et al.*, 2007] 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.
- [Deutsch *et al.*, 2009] 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.
- [Eiter and Gottlob, 1992] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.
- [Fagin *et al.*, 2005] 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.
- [Flouris *et al.*, 2008] Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis, and Grigoris Antoniou. Ontology change: Classification and survey. *Knowledge Engineering Review*, 23(2):117–152, 2008.
- [Ghallab *et al.*, 2004] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning – Theory and Practice*. Elsevier, 2004.
- [Lembo *et al.*, 2010] 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.
- [Limonad *et al.*, 2012] 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.
- [Park, 1976] David Michael Ritchie Park. Finiteness is M-ineffable. *Theoretical Computer Science*, 3(2):173–181, 1976.
- [Poggi *et al.*, 2008] 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.
- [Stirling, 2001] Colin Stirling. *Modal and Temporal Properties of Processes*. Springer, 2001.
- [Vianu, 2009] 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.
- [Winslett, 1990] Marianne Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.