Inconsistency Management in Generalized Knowledge and Action Bases*.

Diego Calvanese, Marco Montali, and Ario Santoso

Free University of Bozen-Bolzano, Bolzano, Italy
lastname@inf.unibz.it

1 Introduction

The combination of static and dynamic aspects in modeling complex organizational domains is a challenging task that has led to study the combination of formalisms from knowledge representation, database theory, and process management [18,23,11]. Specifically, *Knowledge and Action Bases* (KABs) [3] have been put forward recently to provide a semantically rich representation of a domain. In KABs, static aspects are modeled using a Description Logic (DL) [1] knowledge base (KB), while actions are used to evolve its extensional part over time, possibly introducing fresh individuals. An important aspect that has received little attention so far in such systems is the management of *inconsistency* with respect to domain knowledge that may arise when the extensional information is evolved over time. In fact, inconsistency, both in KABs and in related approaches, is typically handled naively by just rejecting updates in actions when they would lead to inconsistency, see e.g., [16,4,9,2].

To overcome this limitation, KABs have been extended lately with mechanisms to handle inconsistency [12]. However, this has been done by defining ad-hoc execution semantics and corresponding ad-hoc verification techniques geared towards specific semantics for inconsistency management. It has also been left open whether adding inconsistency management to the rich setting of KABs, actually increases expressive power. This work attacks these issues by: *(i)* Proposing (standard) GKABs, which enrich KABs with a compact action language inspired by Golog [21] that can be conveniently used to specify processes at a high-level of abstraction. As in KABs, standard GKABs still manage inconsistency naively. *(ii)* Defining a parametric execution semantic for GKABs that is able to elegantly accomodate a plethora of inconsistency-aware semantics based on the well-known notion of repair [17,5,19,13]. *(iii)* Providing several reductions showing that verification of sophisticated first-order temporal properties over inconsistency-aware GKABs can be recast as a corresponding verification problem over standard GKABs. *(iv)* Showing that verification of standard and inconsistency-aware GKABs can be addressed using known techniques, developed for standard KABs.

2 Setting

We use *DL-Lite*$_{\mathcal{A}}$ [8,6] to express KBs, and consider queries such as EQL-Lite(UCQ) (briefly ECQs) [7], to access KBs and extract individuals of interest. To handle inconsis-

* This paper is an abridged version of [14]. Full proofs can be found in [15]

tency in KBs, we follow the *repair-based approaches* in [12], and distinguish between two kinds of approaches: *(i)* those that compute repairs agnostically from the updates (the added/deleted facts) [19,12], among which we have *b-repairs*, which are defined as the maximal (w.r.t. set containment) subsets of an ABox that are consistent with the TBox, and *c-repairs*, which are defined as the intersection of all b-repairs; *(ii)* those that take into account the updates by giving higher priority to the new facts during the repair, as in *bold semantics* for instance-level KB evolution (c.f., [13]).

Here, we consider KABs that are obtained by combining the framework in [3,12] with the action specification formalism in [22], which allows us to have actions that only update an ABox (instead of creating a new ABox at each execution, as in [3,12]). Formally, a KAB is composed by *(i)* a *DL-Lite*$_\mathcal{A}$ TBox T; *(ii)* an initial *DL-Lite*$_\mathcal{A}$ ABox A_0; *(iii)* a finite set Γ of parametric actions that evolve the ABox; *(iv)* a finite set Π of condition-action rules that describe when actions can be executed, and with which parameters. The execution semantics of a KAB is given in terms of a possibly infinite-state *transition system*, whose construction depends on the adopted semantics of inconsistency [12]. As in [12], we call S-KAB a KAB under the standard execution semantics, where inconsistency is naively managed by simply rejecting those updates that lead to an inconsistent state.

To specify temporal properties over KABs, we use the $\mu\mathcal{L}_A^{\text{EQL}}$ logic, the FO variant of μ-calculus defined in [3]. Given a transition system Υ and a closed $\mu\mathcal{L}_A^{\text{EQL}}$ formula Φ, *verification* is the problem of checking whether Φ holds in the initial state of Υ.

3 Golog-KABs and Inconsistency Management

We enrich KABs with a high-level action language inspired by Golog [21]. This allows modelers to represents the dynamics of systems much more compactly. On the other hand, we introduce a parametric execution semantics, which elegantly accommodates the different kinds of inconsistency-aware semantics based on the notion of repair.

A *Golog-KAB (GKAB)* is a tuple $\mathcal{G} = \langle T, A_0, \Gamma, \delta \rangle$, where T, A_0, and Γ are as in standard KABs, and δ is the Golog program characterizing the evolution of the GKAB over time, using the atomic actions in Γ. For simplicity, we only consider a core fragment of Golog based on the action language in [10], and define a *Golog program* as:

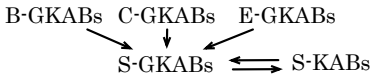
$$\delta ::= \varepsilon \mid \textbf{pick}\ Q(\vec{p}).\alpha(\vec{p}) \mid \delta_1|\delta_2 \mid \delta_1;\delta_2 \mid \textbf{if}\ \varphi\ \textbf{then}\ \delta_1\ \textbf{else}\ \delta_2 \mid \textbf{while}\ \varphi\ \textbf{do}\ \delta$$

where: *(i)* ε is the *empty program*; *(ii)* **pick** $Q(\vec{p}).\alpha(\vec{p})$ is an *atomic action invocation* guarded by an ECQ Q, such that $\alpha \in \Gamma$ is applied by non-deterministically substituting its parameters $\vec{p}$ with an answer of Q; *(iii)* $\delta_1|\delta_2$ is a *non-deterministic choice* between programs; *(iv)* $\delta_1;\delta_2$ is *sequencing*; *(v)* **if** φ **then** δ_1 **else** δ_2, and **while** φ **do** δ are respectively *conditional* and *loop* constructs, using a boolean ECQ φ as condition.

We adopt the functional approach by [20] in defining the semantics of action execution over $\mathcal{G}$, i.e., we assume $\mathcal{G}$ provides two operations: *(i)* ASK, to answer queries over the current KB; *(ii)* TELL, to update the KB through an atomic action. Here the ASK operator corresponds to certain answers computation. The TELL operation is parameterized by *filter relations* f, which are used to refine the way in which an ABox is updated, based on a set of facts to be added and deleted (specified by the action), and we require that the result of the TELL operation is a T-consistent ABox. In this light, filter relations

provide an abstract mechanism to accommodate several inconsistency management approaches in the execution semantics. For instance, we define GKABs with standard execution semantics, briefly S-GKABs, by defining a filter relation f_S that updates an ABox based on the facts to be added and deleted, and does nothing w.r.t. inconsistency (i.e., updates that lead to an inconsistent state are simply rejected). To obtain GKABs with inconsistency-aware semantics, we introduce filter relations f_B, f_C, and f_E, where f_B (resp., f_C) returns a b-repair (resp., c-repair) [12] of the updated ABox, and f_E updates the ABox using the bold semantics of KB evolution [13]. Consecutively, we call the GKABs adopting the execution semantics obtained by employing those filter relations *B-GKABs*, *C-GKABs*, and *E-GKABs*, respectively, and we group them under the umbrella of *inconsistency-aware GKABs (I-GKABs)*.

Verification Results. With respect to verification of $\mu\mathcal{L}_A^{\text{EQL}}$ properties, we have proved the results summarized below, where an arrow indicates that we can reduce verification in (G)KABs in the source to verification in (G)KABs in the target:



To encode S-KABs into S-GKABs, we simulate the standard execution semantics using a Golog program that continues forever to non-deterministically pick an executable action with parameters, or stops if no action is executable. For the opposite direction, the key idea is to inductively interpret a Golog program as a structure consisting of nested processes, suitably composed through the Golog operators. We mark the starting and ending point of each Golog subprogram, and use accessory facts in the ABox to track states corresponding to subprograms. Each subprogram is then inductively translated into a set of actions and condition-action rules encoding its entrance and termination conditions. For all reductions from I-GKABs to S-GKABs, our general strategy is to show that S-GKABs are sufficiently expressive to incorporate the repair-based approaches, so that an action executed under a certain inconsistency semantics can be compiled into a Golog program that applies the action with the standard semantics, and then explicitly handles the inconsistency, if needed.

It is also interesting to observe that the semantic property of *run-boundedness* (which guarantees the decidability of S-KAB verification) [2,3] is preserved by all our reductions. It follows that verification of $\mu\mathcal{L}_A^{\text{EQL}}$ properties over run-bounded GKABs and I-GKABs is decidable, and reducible to standard μ-calculus finite-state model checking.

4 Conclusion

We introduced GKABs, which extend KABs with Golog-inspired high-level programs, and allow for an elegant treatment of inconsistency. We have also shown that verification of rich temporal properties over (inconsistency-aware) GKABs can be recast as verification over standard KABs. Our approach is very general, and can be easily extended to account for other inconsistency handling mechanisms, and more in general data cleaning.

Acknowledgments. This research has been partially supported by the EU IP project Optique (*Scalable End-user Access to Big Data*), grant agreement n. FP7-318338, and by the UNIBZ internal project KENDO (*Knowledge-driven ENterprise Distributed cOmputing*).

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. B. Bagheri Hariri, D. Calvanese, G. De Giacomo, A. Deutsch, and M. Montali. Verification of relational data-centric dynamic systems with external services. In *Proc. of the 32nd ACM SIGACT SIGMOD SIGAI Symp. on Principles of Database Systems (PODS)*, pages 163–174, 2013.
3. B. Bagheri Hariri, D. Calvanese, M. Montali, G. De Giacomo, R. De Masellis, and P. Felli. Description logic Knowledge and Action Bases. *J. of Artificial Intelligence Research*, 46:651–686, 2013.
4. F. Belardinelli, A. Lomuscio, and F. Patrizi. An abstraction technique for the verification of artifact-centric systems. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, pages 319–328, 2012.
5. L. E. Bertossi. Consistent query answering in databases. *SIGMOD Record*, 35(2):68–76, 2006.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodríguez-Muro, and R. Rosati. Ontologies and databases: The *DL-Lite* approach. In S. Tessaris and E. Franconi, editors, *Reasoning Web. Semantic Technologies for Informations Systems – 5th Int. Summer School Tutorial Lectures (RW)*, volume 5689 of *Lecture Notes in Computer Science*, pages 255–356. Springer, 2009.
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. EQL-Lite: Effective first-order query processing in description logics. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 274–279, 2007.
8. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
9. D. Calvanese, G. De Giacomo, D. Lembo, M. Montali, and A. Santoso. Ontology-based governance of data-aware processes. In *Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems (RR)*, volume 7497 of *Lecture Notes in Computer Science*, pages 25–41. Springer, 2012.
10. D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Actions and programs over description logic knowledge bases: A functional approach. In G. Lakemeyer and S. A. McIlraith, editors, *Knowing, Reasoning, and Acting: Essays in Honour of Hector Levesque*. College Publications, 2011.
11. D. Calvanese, G. De Giacomo, and M. Montali. Foundations of data aware process analysis: A database theory perspective. In *Proc. of the 32nd ACM SIGACT SIGMOD SIGAI Symp. on Principles of Database Systems (PODS)*, 2013.
12. D. Calvanese, E. Kharlamov, M. Montali, A. Santoso, and D. Zheleznyakov. Verification of inconsistency-tolerant knowledge and action bases. In *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2013.
13. D. Calvanese, E. Kharlamov, W. Nutt, and D. Zheleznyakov. Evolution of *DL-Lite* knowledge bases. In *Proc. of the 9th Int. Semantic Web Conf. (ISWC)*, volume 6496 of *Lecture Notes in Computer Science*, pages 112–128. Springer, 2010.
14. D. Calvanese, M. Montali, and A. Santoso. Verification of generalized inconsistency-aware knowledge and action bases. In *Proc. of the 24th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2015.
15. D. Calvanese, M. Montali, and A. Santoso. Verification of generalized inconsistency-aware knowledge and action bases (extended version). CoRR Technical Report arXiv:1504.08108, arXiv.org e-Print archive, 2015. Available at `http://arxiv.org/abs/1504.08108`.

16. A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *Proc. of the 12th Int. Conf. on Database Theory (ICDT)*, pages 252–267, 2009.
17. T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.
18. R. Hull. Artifact-centric business process models: Brief survey of research results and challenges. In *Proc. of the 7th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE)*, volume 5332 of *Lecture Notes in Computer Science*, pages 1152–1163. Springer, 2008.
19. D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant semantics for description logics. In *Proc. of the 4th Int. Conf. on Web Reasoning and Rule Systems (RR)*, pages 103–117, 2010.
20. H. J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984.
21. H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *J. of Logic Programming*, 31:59–84, 1997.
22. M. Montali, D. Calvanese, and G. De Giacomo. Verification of data-aware commitment-based multiagent systems. In *Proc. of the 13th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 157–164, 2014.
23. V. Vianu. Automatic verification of database-driven systems: a new frontier. In *Proc. of the 12th Int. Conf. on Database Theory (ICDT)*, pages 1–13, 2009.