Inconsistency Tolerance in OWL 2 QL Knowledge and Action Bases

Statement of Interest

Diego Calvanese, Evgeny Kharlamov, Marco Montali, and Dmitriy Zheleznyakov

KRDB Research Centre, Free University of Bozen-Bolzano, *lastname*@inf.unibz.it

1 From Classical to Data Centric Business Processes

A *Business Process* (*BP*) is constituted by *(i) data* that describes the state of affairs and *(ii)* a set of *activities* (to be performed) over this data. The activities, when combined in what is usually referred to as an *execution flow*, achieve some business goal. When analyzing BPs, one is usually interested in querying possible execution flows to extract useful information and verifying dynamic properties over them.[1] To this effect, BPs are typically modeled via high-level *specifications* of activities [17], which are later compiled into an executable code. Since the logics of business processes is captured by these specifications, tools for querying and analyzing possible execution flows are extremely valuable for companies [11,12,16]. Fully taking into account the presence of data significantly complicates the analysis of execution flows since it makes the system to be an infinite state one in general. On the other hand, it is often assumed that the data is simple enough and does not significantly impact on the analysis of possible execution flows. For these reasons, in the classical modeling paradigm of BPs, the data part is typically abstracted away, and the analysis is carried out under this simplification.

In knowledge-intensive applications, however, where a crucial aspect of BPs is to properly represent the allowed evolutions of the data component, the classical modeling paradigm is not appropriate and one has to take fully into account data in the specification of BPs [14,13,20,1]. For example, in healthcare systems, an electronic record of a patient is the "data" to be manipulated by a process, and "activities" determine how to modify the patient's information (e.g., the registration of patient examination results). In this application scenario it is inappropriate to abstract away from patient records in process specifications. Thus, there is a need for developing and studying formalisms in which both the data component and the process component are first-class citizens. A number of recent proposals follow this approach [9,3,2], which is commonly referred to as *Data-Centric Business Processes* (*DCBPs*).

The dynamic properties one is interested to verify are typically expressed in some variant of temporal logic, such as LTL, CTL, or the (first-order) μ-calculus [18,19], a very expressive temporal logic subsuming most of the other temporal formalisms. In the traditional BP setting, the verification of temporal properties is based on finite-state model checking [8]. However, in DCBPs, the presence of data makes the number of different states of the system potentially infinite. Hence, the verification of dynamic prop-

[1] Note that this task is not trivial since, for instance, a given BP may have a large, possibly infinite, number of possible execution flows.

erties over DCPBs is complicated and represents a significant research challenge. Indeed, neither finite-state model checking nor most of the current techniques for infinite-state model checking [5] can be directly applied to DCBPs, and verification turns out to be undecidable in general [14,20,3].

In the following, we will present a simplified form of a DCPBs specification language that is based on OWL 2 QL and that has been first introduced and studied in [2].

2 (OWL 2 QL) Knowledge and Action Bases

A *Knowledge and Action Base (KAB)* is a triple $\mathcal{K} = (\mathcal{T}, S_0, \Gamma)$, where

(i) $\mathcal{T}$ is a so-called *TBox*, i.e., a set of schema-level constraints representing the intensional-level knowledge about the domain of interest,

(ii) S_0 is a so-called *ABox*, i.e., a set of facts representing the initial information – the *initial state* of the system, and

(iii) Γ is a *set of actions* that specifies how the states of the system should evolve. An *action* $\gamma \in \Gamma$ is a set of effect specifications of the form $q_i(\bar{x}) \rightsquigarrow S_i(\bar{x})$, where $q_i(\bar{x})$ is a query with output variables $\bar{x}$ expressed over (the alphabet of) $\mathcal{T}$, and $S_i(\bar{x})$ is a set of atoms over (the alphabet of) $\mathcal{T}$ and $\bar{x}$. The actions might acquire external information by means of service calls, which are modeled through function symbols.

We illustrate KABs on the following example.

Example 1. Consider the KAB $\mathcal{K}_e = (\mathcal{T}_e, S_0, \Gamma_e)$ with $\Gamma_e = \{\gamma_1, \gamma_2\}$ and

$$\mathcal{T}_e = \{(\mathsf{funct\ marriedTo}),\ \mathsf{De} \sqsubseteq \neg \mathsf{It}\},$$
$$S_0 = \{\mathsf{Married}(\mathsf{Mariano})\},$$
$$\gamma_1 = \mathtt{CA} \cup \{\mathsf{Married}(x) \rightsquigarrow \{\mathsf{marriedTo}(x, \mathbf{roG}(x)),\ \mathsf{De}(\mathbf{roG}(x))\}\},$$
$$\gamma_2 = \mathtt{CA} \cup \{\mathsf{Married}(x) \rightsquigarrow \{\mathsf{marriedTo}(x, \mathbf{roI}(x)),\ \mathsf{It}(\mathbf{roI}(x))\}\}.$$

where CA (which stands for "copy all") is an operator that copies all the atoms of the current state to the new one. Intuitively, $\mathcal{T}_e$ says that a person cannot have more than one spouse (the relation marriedTo is functional) and that Germans are not Italians (and vice-versa); the initial state S_0 states that Mariano is married. Finally, Γ_e says that if a person, say x, is married (i.e., if $\mathsf{Married}(x)$ is satisfied) then one should explicitly add in the new state a fact about *(i)* the marriage of x , i.e., add $\mathsf{marriedTo}(x, \cdot)$ atom to the new state, where the name of his/her spouse can be found via a service call in a registry office in Germany (by calling a service $\mathbf{roG}(x)$ as in γ_1), or in Italy (by calling a service $\mathbf{roI}(x)$ as in γ_2); and *(ii)* the nationality of the x's wife, i.e., the wife is either German, according the action γ_1, or italian, according the action γ_2.

Intuitively, an application of an action γ to a state S returns a new state S' defined as follows. Starting from $S' = \emptyset$, for each action $q_i(\bar{x}) \rightsquigarrow S_i(\bar{x})$ in γ, evaluate q_i over $\mathcal{T} \cup S$ (using the *certain answers* semantics [7]) and add all elements of $S'_i(\bar{a})$ to S' for every $\bar{a} \in cert(q_i, \mathcal{T} \cup S)$, where $cert(q_i, \mathcal{T} \cup S)$ denotes the certain answers of q_i over $\mathcal{T} \cup S$ and $S'_i(\bar{a})$ is the set of ABox assertions resulting from substituting $\bar{x}$ with $\bar{a}$ in $S'_i(\bar{x})$. We refer to [2,4] for details and illustrate the definitions with an example.

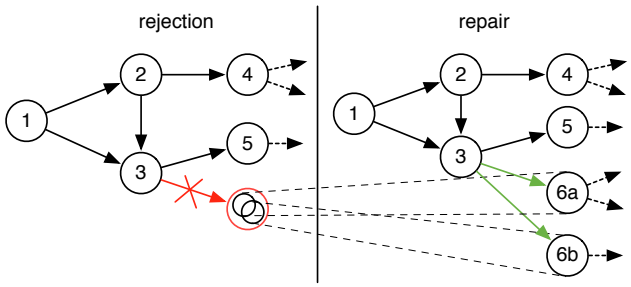

Fig. 1. A possible reject-inconsistency execution flow for Example 1

Example 2. Continuing with Example 1, the following state S_1 can be obtained from S_0 by applying γ_1:

$$S_1 = \{\mathsf{Married}(\mathsf{Mariano}),\ \mathsf{marriedTo}(\mathsf{Mariano}, \mathbf{roG}(\mathsf{Mariano})), \mathsf{De}(\mathbf{roG}(\mathsf{Mariano}))\}.$$

In our study of KABs, we focus on TBoxes expressed in the *DL-Lite* family of description logics [7,6] which forms the logical foundation of OWL 2 QL [10], and allows, in particular, to express functionality of direct and inverse object properties, and equality between constants. Note that, as in OWL 2 QL, and differently from traditional *DL-Lite*, we do not assume the unique name assumption to hold.

3 Execution Flow of KABs

The execution flow associated with a KAB $\mathcal{K} = (\mathcal{T}, S_0, \Gamma)$ is a graph $\mathcal{G}(\mathcal{K})$ whose nodes are states, reachable from S_0 by "executing" actions from Γ, and that are consistent with $\mathcal{T}$. More formally, (S, S') is an edge of $\mathcal{G}(\mathcal{K})$ if and only if *(i)* S' can be obtained from S by applying some action $\gamma \in \Gamma$, and *(ii)* $S' \cup \mathcal{T}$ is a consistent KB. In Figure 1, left, there is a (fragment of) $\mathcal{G}(\mathcal{K}_e)$ for $\mathcal{K}_e$ of Example 1. State 1 is initial and States 2-6 are reachable from it. Here we assume that States 2-5 are consistent with $\mathcal{T}_e$ and State 6 is not; thus, State 6 is "rejected" from $\mathcal{G}(\mathcal{K}_e)$ and there is no edge from State 3 to 6.

Example 3. Continuing with Example 2, the application of γ_2 to S_1 yields the state

$$S_2 = S_1 \cup \{\mathsf{marriedTo}(\mathsf{Mariano}, \mathbf{roI}(\mathsf{Mariano})),\ \mathsf{It}(\mathbf{roI}(\mathsf{Mariano}))\}.$$

We have that $S_2 \cup \mathcal{T}$ is inconsistent and hence S_2 is rejected from $\mathcal{G}(\mathcal{K}_e)$. The inconsistency comes from the fact that S_2 includes both $\mathsf{De}(\mathbf{roG}(\mathsf{Mariano}))$ and $\mathsf{It}(\mathbf{roI}(\mathsf{Mariano}))$, which together with $\mathsf{De} \sqsubseteq \neg\mathsf{It} \in \mathcal{T}_e$ leads to $\mathbf{roG}(\mathsf{Mariano}) \neq \mathbf{roI}(\mathsf{Mariano})$. At the same time, S_2 contains both $\mathsf{marriedTo}(\mathsf{Mariano}, \mathbf{roG}(\mathsf{Mariano}))$ and $\mathsf{marriedTo}(\mathsf{Mariano}, \mathbf{roI}(\mathsf{Mariano}))$, and due to functionality of $\mathsf{marriedTo}$ in $\mathcal{T}_e$ this yields that $\mathbf{roG}(\mathsf{Mariano}) = \mathbf{roI}(\mathsf{Mariano})$.

In [4,2] it was shown that even checking (simple) propositional LTL safety properties over execution flows represented by $\mathcal{G}(\mathcal{K})$ is in general undecidable. It was also shown that a specific form of weak-acyclicity condition on the action specification of KABs (inspired by weak-acyclicity in data-exchange [15]), is sufficient to guarantee

decidability. We argue here, however, that the way in which $\mathcal{G}(\mathcal{K})$ is defined in [4,2] is too restrictive, since for many applications it is desirable not to immediately reject states that are inconsistent with $\mathcal{T}$. Indeed, the inconsistency may be due to a possibly very small portion of the ABox, so in this case one might want to allow for the action generating the inconsistent state to be executed, while "repairing" the generated inconsistency. Consider also that inconsistencies may arise from the information brought into the state by calls to external services. These are out of the control of the system, so that conflicts of knowledge may be unavoidable when the external information is *integrated* with the one coming from the state in which the action was applied.

4 Repairing Inconsistent States

In Example 3, the state S_2 is rejected because some person would have to be both Italian and German, which contradicts the TBox. However, the inconsistency is caused only by a (small) portion of the ABox, and therefore it would be desirable not to lose the remaining consistent part and keep this state, *repairing* it beforehand. Indeed, it could be the case that the wife of Mariano used to be an Italian, but then she moved to Germany and changed her citizenship (or the other way around), while the information in the registry offices has not been updated. We do not have control over the offices, but we can repair the state correspondingly to the options we have, trying to remove the inconsistency while keeping at the same time as much information as possible.

Example 4. The following states are possible repairs of S_2 from Example 3:

$$S^1_{rep} = \{\mathbf{roG}(\mathsf{Mariano}) = \mathbf{roI}(\mathsf{Mariano}),\ \mathsf{Married}(\mathsf{Mariano}),\ \mathsf{De}(\mathbf{roG}(\mathsf{Mariano})),\ \mathsf{marriedTo}(\mathsf{Mariano}, \mathbf{roG}(\mathsf{Mariano}))\},$$

$$S^2_{rep} = \{\mathbf{roG}(\mathsf{Mariano}) = \mathbf{roI}(\mathsf{Mariano}),\ \mathsf{Married}(\mathsf{Mariano}),\ \mathsf{It}(\mathbf{roI}(\mathsf{Mariano})),\ \mathsf{marriedTo}(\mathsf{Mariano}, \mathbf{roI}(\mathsf{Mariano}))\}.$$

The repair S^1_{rep} represents the case when Mariano's wife is German, and S^2_{rep} – Italian.

In Figure 1, right, we present a *repair-based* execution flow, where instead of rejecting the inconsistent State 6, we set as its successors its repairs, namely State 6a and State 6b, and continue to execute Γ.

5 Our Goals

We are currently working on various aspects related to the specification of DCBPs and KABs and the verification of temporal properties over them. The specific aspect that we have discussed here, namely the adoption of a repair-based semantics to deal with the inconsistencies arising in a state, is a particularly challenging direction. In particular, we are interested in investigating how to extend to this new setting the decidability results established for verification of μ-calculus properties over DCBPs [3,4] and KABs [2].

This work is carried out within the EU project ACSI[2], within which we are also studying uses-cases for KABs and implementing prototype verification tools.

[2] `http://www.acsi-project.eu/`

Acknowledgements. This research has been partially supported by the EU under the ICT Collaborative Project ACSI (Artifact-Centric Service Interoperation), grant agreement n. FP7-257593.

References

1. S. Abiteboul, L. Segoufin, and V. Vianu. Modeling and verifying Active XML artifacts. *IEEE Bull. on Data Engineering*, 32(3):10–15, 2009.
2. B. Bagheri Hariri, D. Calvanese, G. De Giacomo, and R. De Masellis. Verification of conjunctive-query based semantic artifacts. In *Proc. of DL 2011*, volume 745 of *CEUR,* `ceur-ws.org`, 2011.
3. B. Bagheri Hariri, D. Calvanese, G. De Giacomo, R. De Masellis, and P. Felli. Foundations of relational artifacts verification. In *Proc. of BPM 2011*, LNCS. Springer, 2011.
4. B. Bagheri Hariri, D. Calvanese, G. De Giacomo, A. Deutsch, and M. Montali. Verification of relational data-centric dynamic systems with external services. CoRR Technical Report arXiv:1203.0024, arXiv.org e-Print archive, 2012. Available at `http://arxiv.org/abs/1203.0024`.
5. O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification of infinite structures. In *Handbook of Process Algebra*. Elsevier Science, 2001.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Linking data to ontologies: The description logic *DL-Lite*$_A$. In *Proc. of OWLED 2006*, volume 216 of *CEUR,* `ceur-ws.org`, 2006.
7. 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.
8. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. The MIT Press, Cambridge, MA, USA, 1999.
9. D. Cohn and R. Hull. Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Bull. on Data Engineering*, 32(3):3–9, 2009.
10. B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *J. of Web Semantics*, 6(4):309–322, 2008.
11. D. Deutch. Querying probabilistic business processes for sub-flows. In *Proc. of ICDT 2011*, pages 54–65, 2011.
12. D. Deutch and T. Milo. Type inference and type checking for queries on execution traces. *PVLDB*, 1(1):352–363, 2008.
13. A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *Proc. of ICDT 2009*, pages 252–267, 2009.
14. A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven web applications. *J. of Computer and System Sciences*, 73(3):442–474, 2007.
15. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.
16. R. Hull and J. Su. Tools for composite web services: a short overview. *SIGMOD Record*, 34(2):86–95, 2005.
17. IBM. Business Process Execution Language for Web Services, 2002. Available at `http://www.ibm.com/developerworks/library/ws-bpel/`.
18. D. Park. Finiteness is mu-ineffable. *Theoretical Computer Science*, 3:173–181, 1976.
19. C. Stirling. *Modal and Temporal Properties of Processes*. Springer, 2001.
20. V. Vianu. Automatic verification of database-driven systems: a new frontier. In *Proc. of ICDT 2009*, pages 1–13, 2009.