Supporting Ontology-Mediated Stream Reasoning with Model Checking

Clemens Dubslaff, Patrick Koopmann, and Anni-Yasmin Turhan

Technische Universität Dresden, Dresden, Germany
{clemens.dubslaff,patrick.koopmann,anni-yasmin.turhan}@tu-dresden.de

Abstract. Many stream reasoning applications make use of ontology reasoning to deal with incomplete data. To this end, definitions of complex concepts and elaborate ontologies are already used in event descriptions and queries. On the other hand, model checking can support stream reasoning, e.g., by runtime verification to predict future events or by classical offline verification. For such a combined use of model checking and stream reasoning, it is crucial that events are modeled in a compatible way. We have recently introduced and implemented a framework for ontology-mediated probabilistic model checking, which would allow to use the same ontology and event descriptions in both: the stream reasoner and the model checker. In this short paper we present this framework and discuss and motivate its use in the context of stream reasoning.

A core task of stream reasoning is to detect complex events occurring in data from a collection of different data sources that reflect the behavior of an observed system over time. The data sources need to be integrated to a certain extent and often give only incomplete information on the observed system's state. Ontology-mediated query answering (OMQA) [9] addresses these two arising challenges and has often been applied to stream reasoning scenarios [8,16,4]. In OMQA the raw data collected in a database gets augmented by the background information of a Description Logics (DL) knowledge base. While the actual ontology is often given in a DL from the OWL 2.0 standard, complex situations to be recognized are specified as queries in a dedicated query language. The task of situation recognition then boils down to answer the query over the DL knowledge base. A variety of query languages have been developed, e.g., for temporal data [10,2,3], probabilistic temporal data [17], and incorporating numerical data [5,1].

Probabilistic model checking (PMC) [6,14] is an automated verification technique that has been successfully applied in many areas to verify non-functional requirements such as energy efficiency or reliability. Most prominently, PMC is applied during design time on an operational system model expressed, e.g., by probabilistic guarded commands (PGCs), the input language of the PMC tool Prism [18]. PMC techniques can also be used in runtime verification [19,15] where monitors predict the satisfaction of requirements based on execution histories of the monitored system.

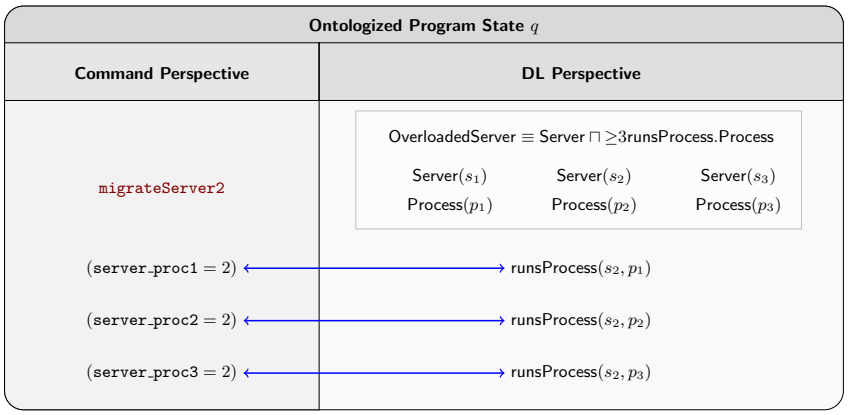

Fig. 1. Perspectives on the ontologized program state q: the command perspective in PRISM (left) and DL perspective (right), linked by the interface (arrows).

Ontology-mediated PMC

OMQA and PMC are both well-investigated logic-based approaches to reason about system properties with a wide range of optimized and matured tools. However, they serve different purposes: while OMQA mainly reasons about knowledge-intensive systems, PMC focuses on operational system models and their behavioral properties. To integrate the advantages of both approaches, we proposed *ontology-mediated PMC* (OM-PMC) in [12,13], enabling a verification of operational systems models in knowledge-intensive contexts. The key idea for the integration is an interface between the two specification formalisms for OMQA and PMC, which allows a clear separation of concerns. The interface then provides a mediation between the formalisms by means of:

1. A mapping from system states described in the PRISM language to a DL ontology, i.e., to a set of TBox axioms and assertions. Effectively, each state q described in PRISM is associated with an ontology $\mathcal{O}_q$, see Figure 1.
2. So-called *hooks*, which are special statements in PRISM that get their truth value as the result of a Boolean query over the state's ontology $\mathcal{O}_q$ returned from the DL component (i.e., answered by a DL reasoner).

To this end, PMC may refer to DL queries via hooks and thus includes DL reasoning results in the model-checking process. As the selection of axioms in $\mathcal{O}_q$ for each state q is determined before the model-checking process takes place, the answers to the corresponding hooks over $\mathcal{O}_q$ can be computed beforehand and "re-written" into the modeling language of PRISM. This kind of rewriting to the standard modeling language of PRISM was implemented and evaluated in a prototype. Our initial results provided in [12,13] suggest that OM-PMC can help to mitigate the notorious state-explosion problem of PMC by restricting the state space through precise contextual information in the DL knowledge base.

Supporting Stream Reasoning

PMC can well be used to reason about streams: naturally through runtime-verification techniques where system properties are predicted based on observed stream events, but also in an offline verification where a predictive model of the streams is included in the system's specification [20]. We propose to support stream reasoning by OM-PMC, where stream events influence the system's behavior through hooks evaluated by OMQA as in classical stream reasoning.

Runtime Verification. For OM-PMC-supported runtime verification, we have to provide an operational model that includes runtime hook evaluations. As hook evaluations rely on the assigned query over the knowledge base that is also depending on stream events, we hence model them nondeterministically. While this leads to an exponential blow-up in the number of hooks, a family-based verification approach [11] can benefit from the commonalities in the systems behavior specification and thus enable PMC for all combinations of hook satisfactions. The PMC results are then stored for all state-hook pairs, e.g., providing the energy consumption in best- and worst-case scenarios with the respective hooks satisfied. During runtime verification, our approach is then to re-evaluate hooks based on stream events, determine corresponding states in the systems model, and then either look up the PMC results for sophisticated requirements or eventually refine them through additional analysis of the systems' model. The latter amounts to standard runtime verification and may also include assumptions and predictions on future stream events.

Offline Verification. To reason about streams during design time of adaptive systems, we rely on a model of the stream events to happen during execution time. Here, a purely non-deterministic modeling of stream events as proposed in the last paragraph is usually too coarse towards meaningful analysis results. Based on stochastic models obtained from statistical evaluation of stream data that reflects the standard application scenarios [20,7], OM-PMC can be used for a quantitative analysis of streams. For instance, stream events could represent bandwidth requirements in server systems where a hook is assigned to high bandwidth requirements usually occurring during daytime. Based on the assumption of day-night shifts, e.g., relying on DL reasoning, a convolution of bandwidth requirements from server logs leads a stream event model [7].

Thus, OM-PMC results can be used as additional data source, supply predictions on the likelihood of coming events based on statistical models, and support decision making through monitoring and runtime verification.

Acknowledgements. The authors are supported by the DFG through the Collaborative Research Center TRR 248 (see `https://perspicuous-computing.science`, project ID 389792660), the Cluster of Excellence EXC 2050/1 (CeTI, project ID 390696704, as part of Germany's Excellence Strategy), and the Research Training Groups QuantLA (GRK 1763) and RoSI (GRK 1907).

References

1. Alrabbaa, C., Koopmann, P., Turhan, A.Y.: Practical query rewriting for DL-Lite with numerical predicates. In: Proc. of the 5th Glob. Conf. on AI (GCAI'19) (2019)
2. Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: Ontology-mediated query answering over temporal data: A survey. In: 24th Int. Sympos. on Temp. Representation and Reasoning, TIME (2017)
3. Baader, F., Borgwardt, S., Koopmann, P., Ozaki, A., Thost, V.: Metric temporal description logics with interval-rigid names. ACM Transactions on Computational Logic **21**(4), 30:1–30:46 (2020)
4. Baader, F., Borgwardt, S., Koopmann, P., Thost, V., Turhan, A.Y.: Semantic technologies for situation awareness. KI – Künstliche Intelligenz (2020)
5. Baader, F., Koopmann, P., Turhan, A.Y.: Using ontologies to query probabilistic numerical data. In: Frontiers of Combining Systems (FroCoS). LNCS (2017)
6. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
7. Baier, C., Dubslaff, C.: From verification to synthesis under cost-utility constraints. ACM SIGLOG News **5**(4), 26–46 (2018)
8. Beck, H., Dao-Tran, M., Eiter, T.: LARS: A logic-based framework for analytic reasoning over streams. Artif. Intell. **261**, 16–70 (2018)
9. Bienvenu, M.: Ontology-mediated query answering: Harnessing knowledge to get more from data. In: Proc. of the 25th Int. Joint Conf. on AI, IJCAI 2016 (2016)
10. Borgwardt, S., Lippmann, M., Thost, V.: Temporalizing rewritable query languages over knowledge bases. Journal of Web Semantics **33**, 50–70 (Aug 2015)
11. Dubslaff, C., Baier, C., Klüppelholz, S.: Probabilistic model checking for feature-oriented systems. Trans. on Aspect-Oriented Software Dev. **12**, 180–220 (2015)
12. Dubslaff, C., Koopmann, P., Turhan, A.Y.: Ontology-mediated probabilistic model checking. In: Proc. of the 15th International Conference on Integrated Formal Methods (iFM'19). LNCS, vol. 11918, pp. 194–211. Springer (2019)
13. Dubslaff, C., Koopmann, P., Turhan, A.Y.: Enhancing probabilistic model checking with ontologies. Formal Aspects of Computing (2021)
14. Forejt, V., Kwiatkowska, M.Z., Norman, G., Parker, D.: Automated verification techniques for probabilistic systems. In: Proc. of the School on Formal Methods for the Design of Computer, Communication and Software Systems, Formal Methods for Eternal Networked Software Systems (SFM'11) (2011)
15. Forejt, V., Kwiatkowska, M.Z., Parker, D., Qu, H., Ujma, M.: Incremental runtime verification of probabilistic systems. In: Proc. of the 3rd International Conference on Runtime Verification (RV). vol. 7687 (2013)
16. Kharlamov, E., Kotidis, Y., Mailis, T., Neuenstadt, C., Nikolaou, C., Özçep, Ö.L., Svingos, C., Zheleznyakov, D., Ioannidis, Y.E., Lamparter, S., Möller, R., Waaler, A.: An ontology-mediated analytics-aware approach to support monitoring and diagnostics of static and streaming data. J. Web Semant. **56**, 30–55 (2019)
17. Koopmann, P.: Ontology-based query answering for probabilistic temporal data. In: Proc. of the 33rd AAAI Conf. on AI (AAAI'19) (2019)
18. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Proc. of the 23rd Intern. Conf. on Computer Aided Verification (CAV'11). LNCS, vol. 6806, pp. 585–591 (2011)
19. Leucker, M., Schallhart, C.: A brief account of runtime verification. The Journal of Logic and Algebraic Programming **78**(5), 293–303 (2009)
20. Tiger, M., Heintz, F.: Stream reasoning using temporal logic and predictive probabilistic state models. In: 2016 23rd International Symposium on Temporal Representation and Reasoning (TIME). pp. 196–205 (2016)