On Implementing Temporal Query Answering in *DL-Lite*[*] (extended abstract)

Veronika Thost[1], Jan Holste[2], and Özgür Özçep[3]

[1] Technische Universität Dresden, Germany, thost@tcs.inf.tu-dresden.de
[2] Technische Universität Hamburg-Harburg, Germany, mail@janholste.com
[3] University of Lübeck, Germany, oezcep@ifis.uni-luebeck.de

Temporal information plays a central role in many applications of ontology-based data access (OBDA). For example, knowledge about the past is usually kept in patient records, and collected by companies or scientific projects as MesoWest[4], focusing on weather data. Such applications obviously benefit from using ontologies for data integration and access (e.g., the wind force 'Storm' on the well-known Beaufort Wind Force Scale is equally characterized by wind speed and wave height, which can be represented by a general concept inclusion as $\mathsf{HighWindSpeed} \sqcup \mathsf{HighWaves} \sqsubseteq \mathsf{Storm}$). Temporal knowledge is however not taken into account by systems implementing OBDA, in general. Though, assuming that we consider several weather stations' data of the past 24 hours, a query such as the following could be interesting: *"Get the heritage sites that are nearby a weather station, for which at some time in the past (24 hours) a danger of a hurricane was detected, since then, the wind force has been continuously very high, and it increased considerably during the two latest times of observation."*

For that reason, we investigate different approaches for answering *temporal conjunctive queries* (TCQs) [4, 5] w.r.t. ontologies written in the description logic (DL) *DL-Lite*$_{core}$ (hereinafter called *DL-Lite*). TCQs combine conjunctive queries (CQs) via LTL operators[5] and have already been studied extensively in the context of *DL-Lite* [13, 8]. The above example query could be specified as the following TCQ:

$$\mathsf{HeritageSite}(x) \land \mathsf{WeatherStation}(y) \land \mathsf{nearby}(x, y) \land$$
$$\big(\mathsf{HighWind}(y)\,\mathsf{S}\,\mathsf{DangerOfHurricane}(y)\big) \land \bigcirc^{-}\,\mathsf{Storm}(y) \land \mathsf{ViolentStorm}(y),$$

asking for all pairs (x, y) of heritage sites and nearby weather stations, whose sensor values at some point in time implied a danger of a hurricane, *since* (S) then, the measurements have implied Beaufort category 'high wind', in the *previous* ($\bigcirc^{-}$) moment of observation they implied category 'storm', and the latest values imply 'violent storm'. The semantics of TCQs is based on temporal knowledge bases, which, in addition to the ontology (assumed to hold at every point in time), contain a sequence of fact bases $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n$, representing the data collected at specific points in time. Especially note that the ontology and the fact bases itself are formulated in a classical DL.

Related Work

On the DL side, there are various optimized systems realizing OBDA [10]. In particular, the so-called *rewriting approach* realized by Ontop [15] allows for efficient query

[*]Partially supported by the DFG in CRC 912 (HAEC) and by the EU Commission as part of the FP7 project Optique.

[4]http://mesowest.utah.edu/

[5]Please note that we do not consider negation.

answering w.r.t. an ontology written in *DL-Lite*. Specifically, Ontop internally rewrites a given conjunctive query, which is written in the abstract vocabulary of the ontology, into an SQL query that encodes the relevant ontological knowledge but addresses a standard database system; the latter can then be used to store the data and efficiently answer the queries. But whereas a lot of DL research is studying temporal extensions of ontology and query languages [2, 6, 3, 11, 13, 5, 8, 14], none of the freely available systems takes the temporal nature of the data into account, yet (i.e., the query languages supported do not provide operators for explicitly referencing different points in time). Nevertheless, recently, several practical approaches for answering temporal queries have been developed in the fields of (RDF) stream reasoning [7] and complex event processing [9, 1]. These systems are tailored to continuously answering given queries over an infinite stream of data—which is usually realized by restricting the focus to a *window* of the data (i.e., instead of considering all the past data available, the number of considered time points or data instances is fix). But only few of these systems support ontologies, yet, and standards for a common stream representation and processing, for query languages, and for operation semantics are still to be developed.

Our work complements these applied approaches by starting from a DL perspective, where many use cases consider static data, ontologies are important, and there are several well-investigated query languages. Specifically, we study three pragmatic approaches for answering TCQs based on the work of [8]. We prototypically implemented and evaluated them, and in this paper report on our experiences.

Algorithms for Temporal Query Answering

In particular, [8] propose algorithms for answering temporal queries (w.r.t. TKBs) that generalize TCQs in that they combine queries of a generic atemporal query language $\mathcal{Q}$ via LTL-operators (i.e., we restrict $\mathcal{Q}$ to CQs).[6] Similar to the streaming scenario, [8] assume a fix set of TCQs to be answered continuously at time point n.

We first implemented the Iterative Algorithm (IA) (cf. Section 6 in [8]), which iteratively computes sets of answers to several subqueries of the TCQ to be answered, for each time point i, $0 \leq i \leq n$. For example, the answers to $\bigcirc^{-}\mathsf{Storm}(x)$ at i are obtained by evaluating $\mathsf{Storm}(x)$ at $i-1$. Since the processing at i only uses $\mathcal{A}_i$ and the sets computed for the previous moment, whose sizes are bounded, the IA achieves a so-called *bounded history encoding* (i.e., it's runtime does not depend on the number of considered fact bases). A growing number of data from the past however usually leads to an increase in processing time, in practice. We therefore describe a window-based variant of the IA, the Vector Algorithm (VA) (cf. Section 4 in [12]). With this approach, the entire history can still be regarded (i.e., by considering it as one window) but, for example, streaming scenarios can be managed, too. The VA specifically supports *sliding* windows, where the TCQs are not evaluated at every time point, but in fix intervals, by only regarding the then necessary of the above mentioned sets of answers. To answer the query $\bigcirc^{-}\mathsf{Storm}(x)$ at every second point in time, for example, $\mathsf{Storm}(x)$ does not always have to be evaluated. Our implementations of both algorithms are based on materializing the fact bases and thus may be extended in the future w.r.t. other rewritable

[6][8] assume $\mathcal{Q}$ to be a *rewritable* query language, which basically means that the certain answers to every query in $\mathcal{Q}$ w.r.t. a knowledge base $\mathcal{K}$ can be obtained by answering an appropriate adaptation of the query in a canonical model of $\mathcal{K}$.

query languages (e.g., by employing a corresponding reasoner for the atemporal query answering, temporal queries over $\mathcal{EL}$-TKBs could be answered).

To get an impression of the performance of a temporal rewriting approach, we finally consider (a rather simple) Rewriting Algorithm (RA) translating TCQs into standard database queries. Note that the idea is similarly described by [8]. However, in [8], the TCQs are assumed to be rewritten into a *temporal* standard query language. Since it turned out that such query languages are only partially supported by existing databases, we describe a rewriting into basic SQL. In particular, we developed the QuAnTOn library, which translates TCQs w.r.t. a *DL-Lite* ontology into SQL queries encoding the relevant ontological knowledge and addressing a standard database. QuAnTOn applies Ontop for rewriting the CQs into SQL, appropriately combines these SQL queries, and adds clauses specifying the temporal conditions. For example, the TCQ $\bigcirc^{-}\mathsf{Storm}(x)$ could be translated into the below SQL query with the inner query coming from Ontop.

```
SELECT loc FROM
(SELECT loc, sensor, timestamp FROM sensorvalues
       WHERE (sensor = 'wind-speed' AND value > 24.5) OR
              (sensor = 'wave-height' AND value > 9))
WHERE timestamp = current_time-1;
```

Evaluation Results

We focus on the querying of weather data as outlined above and regard sequences of fact bases each containing 1000 assertions representing sensor measurements. We chose this (streaming) scenario, sketched in [17], to facilitate a comparison with stream reasoning systems, which is planned for future work. We also use an extension of the corresponding sensor-observation ontology[7]. To especially learn about the performance of different kinds of TCQs, we only use very simple CQs within the latter. We investigate the time needed for rewriting and answering TCQs in dependence of the number $n + 1$ of fact bases considered and show the below;[8] the full details can be found in [16, 12].

- The time required for the initial preprocessing (per TCQ) is negligible: less than 1 ms for the IA/VA and about 200 ms for the application of the RA.
- The scenario considering a fix set of sensors suits our constant domain assumption, and the IA shows a rather constant performance (Figure 1).
- The VA is only applicable for windows of moderate size, due to memory issues.
- The time taken for answering the queries of the RA strongly depends on the kind of the queries and the application (e.g., the impact of fetching the results is large).

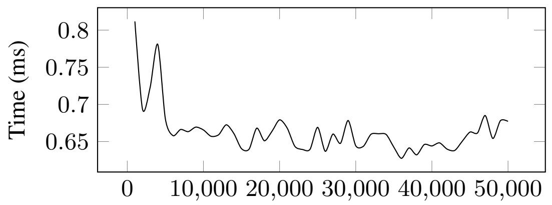

Fig. 1. The time the IA takes for answering a TCQ in dependence of the number of fact bases, $n+1$.

[7]http://wiki.knoesis.org/index.php/LinkedSensorData

[8]The tests were run on an 2,2 GHz Intel Core i7 machine with 4 GB RAM. For answering the queries returned by QuAnTOn, we applied a MySQL (v5.2.38) database.

References

1. Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N.: Stream reasoning and complex event processing in etalis. Semant. web 3(4), 397–407 (Oct 2012)
2. Artale, A., Kontchakov, R., Lutz, C., Wolter, F., Zakharyaschev, M.: Temporalising tractable description logics. In: Goranko, V., Wang, X.S. (eds.) Proceedings of the 14th International Symposium on Temporal Representation and Reasoning (TIME 2007), pp. 11–22. IEEE Press (2007)
3. Artale, A., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: A cookbook for temporal conceptual data modelling with description logics. ACM Transactions on Computational Logic 15(3), 25 (2014)
4. Baader, F., Borgwardt, S., Lippmann, M.: Temporalizing ontology-based data access. In: Bonacina, M.P. (ed.) Proc. of the 24th Int. Conf. on Automated Deduction (CADE'13). Lecture Notes in Computer Science, vol. 7898, pp. 330–344. Springer-Verlag (2013)
5. Baader, F., Borgwardt, S., Lippmann, M.: Temporal query entailment in the description logic $\mathcal{SHQ}$. Journal of Web Semantics (2015)
6. Baader, F., Ghilardi, S., Lutz, C.: LTL over description logic axioms. ACM Transactions on Computational Logic 13(3), 21:1–21:32 (2012)
7. Balduini, M., Calbimonte, J.P., Corcho, O., Dell'Aglio, D., Della Valle, E.: Stream reasoning for linked data, `http://streamreasoning.org/events/sr4ld2014`
8. Borgwardt, S., Lippmann, M., Thost, V.: Temporalizing rewritable query languages over knowledge bases. Journal of Web Semantics (2015), in press.
9. Cugola, G., Margara, A.: Tesla: A formally defined event specification language. In: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems (DEBS '10). pp. 50–61. ACM, New York, NY, USA (2010)
10. Gonçalves, R.S., Bail, S., Jimenez-ruiz, E., Parsia, B., Glimm, B., Kazakov, Y.: OWL Reasoner Evaluation (ORE) workshop 2013 results: Short report
11. Gutiérrez-Basulto, V., Jung, J.C., Schneider, T.: Lightweight description logics and branching time: A troublesome marriage. In: Proc. of the 14th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'14). AAAI Press (2014)
12. Holste, J.: Ontology-Based Temporal Reasoning on Streams. Master's thesis, Hamburg University of Technology (October 2014), `http://www.sts.tuhh.de/pw-and-m-theses/2014/holste14a.pdf`
13. Klarman, S., Meyer, T.: Querying temporal databases via OWL 2 QL. In: Kontchakov, R., Mugnier, M.L. (eds.) Web Reasoning and Rule Systems, Lecture Notes in Computer Science, vol. 8741, pp. 92–107. Springer International Publishing (2014)
14. Özçep, O., Möller, R., Neuenstadt, C.: A stream-temporal query language for ontology based data access. In: Lutz, C., Thielscher, M. (eds.) KI 2014: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 8736, pp. 183–194. Springer International Publishing (2014)
15. Rodriguez-Muro, M., Calvanese, D.: High performance query answering over *DL-Lite* ontologies. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012 (2012)
16. Thost, V., Holste, J., Özgür Özçep: On implementing temporal query answering in *DL-Lite*. LTCS-Report 15-12, Chair for Automata Theory, TU Dresden, Germany (2015), `http://lat.inf.tu-dresden.de/research/reports/2015/THO-LTCS-15-12.pdf`, see http://lat.inf.tu-dresden.de/research/reports.html.
17. Zhang, Y., Duc, P.M., Corcho, O., Calbimonte, J.P.: SRBench: a streaming RDF/SPARQL benchmark. In: Proceedings of the 11th International Conference on The Semantic Web - Volume Part I. pp. 641–657. ISWC'12, Springer-Verlag, Berlin, Heidelberg (2012)