Datalog with Existential Quantifiers and Temporal Operators (Extended Abstract)

Matthias Lanzinger[1], Przemysław A. Wałęga[1]

[1]University of Oxford, Department of Computer Science, Wolfson Building, Parks Road, Oxford OX1 3QD, United Kingdom

1. Introduction

We report on our ongoing research on extending Datalog with existential rules and metric temporal operators. Such an extension would be of high practical importance; however, as we show, many commonly studied syntactic restrictions yielding decidable fragments of Datalog with existential rules (known as Datalog$^{\exists}$, Datalog$^{\pm}$, or tuple-generating dependencies TGDs) are not applicable in the temporal setting. Indeed, most of these restrictions are either too weak to guarantee decidability or cannot be naturally extended to the our setting. In turn, we propose to combine syntactic restrictions with semantic modifications as a possible path towards decidable reasoning, and present first complexity results in this novel direction. Moreover, we discuss promising alternative properties under which efficient reasoning in the presence of existential quantification and temporal operators may be possible.

Our main aim is to extend DatalogMTL [1]—a temporal Datalog with operators from metric temporal logic MTL [2] interpreted over the rational time line—with existential rules while, at the same time, preserving decidability of reasoning. DatalogMTL is a highly expressive language with a number applications, for example, in stream reasoning [3] and temporal ontology-based query answering [4, 5]. By allowing for MTL operators in Datalog atoms, it allows us to write atoms such as $\diamondminus_{1,4}A(x,y)$, which states that the atom $A(x,y)$ did hold at some past time point which is at least 1 and at most 4 seconds ago, whereas $\boxminus_{1,4}A(x,y)$ states that $A(x,y)$ did hold continuously during the above mentioned interval of time.

We propose an extension, DatalogMTL$^{\exists}$, of DatalogMTL with existential rules, which allows us to use existential quantifiers in rule heads. For example, it allows us for writing a rule

$$\exists y \boxminus_{[10,10]} \mathit{NobelIn}(x,y) \leftarrow \mathit{Nobel10thAniv}(x),$$

which states that if x celebrates the 10th anniversary of receiving the Nobel prize, there needs to exist a category y in which, 10 years ago, x received the Nobel prize. The access to both MTL operators and existential rules, provides a powerful extension of Datalog. The main obstacle, as

Datalog 2.0 2022: 4th International Workshop on the Resurgence of Datalog in Academia and Industry, September 05, 2022, Genova - Nervi, Italy
matthias.lanzinger@cs.ox.ac.uk (M. Lanzinger); przemyslaw.walega@cs.ox.ac.uk (P. A. Wałęga)
0000-0002-7601-3727 (M. Lanzinger); 0000-0003-2922-0472 (P. A. Wałęga)

we show next, is that combining temporal operators and existential rules quickly leads to bad computational behaviour. Consequently, it raises a question how to construct decidable (and preferably low-complexity) variants of DatalogMTL$^{\exists}$.

2. Our Results on DatalogMTL$^{\exists}$

In this section, we give a summary of our ongoing research on the complexity of reasoning in DatalogMTL$^{\exists}$ and in its modifications. We also discuss some conceptual challenges which disallow for using standard reasoning techniques in existential rules.

Our so-far obtained results are summarised in Table 1, where we consider DatalogMTL$^{\exists}$ under the *natural* and *uniform* semantics, separately, as well as standard syntactical restrictions, constituted by guarded and weakly acyclic programs. By the natural semantics we mean that whenever an existential rule 'fires' it can invent arbitrary new constants. Hence, the same ground existential rule can invent different constants when fired in different time points; this can lead to undesired meaning of existential rules and also to bad computational behaviour—since we consider an infinite, and moreover dense, rational time line. In contrast, by the uniform semantics, we mean that a ground existential rule invents the same constants in all time points it fires; in this sense applications of existential rules are 'uniform'. Moreover, together with standard open-world assumption (OWA), which allows for new values invention by existential rules, we consider also closed-world assumption (CWA) which cannot invent new constants. Note that existential quantification under CWA can be seen as a disjunction over all explicitly mentioned constants in the program, so it is less interesting from the conceptual point of view, but in practice can be useful for the concise representation of huge disjunctions.

		Full DatalogMTL$^{\exists}$	Guarded programs	Weakly acyclic programs
Natural semantics	OWA	undecidable	undecidable	undecidable
	CWA	undecidable		
Uniform semantics	OWA	undecidable	undecidable	**2-ExpSpace-co.**
	CWA	**ExpSpace-co.**		

Table 1
Fact entailment and consistency checking complexity in DatalogMTL$^{\exists}$

Regarding the results in Table 1, we observe that under the natural semantics, undecidability of the full DatalogMTL$^{\exists}$ follows immediately from the well-known undecidability of Datalog$^{\exists}$. However, we additionally show that the guarded and weakly acyclic fragments of DatalogMTL are already undecidable, which is in contrast to Datalog$^{\exists}$. Notably, even the CWA case is undecidable under natural semantics (even for programs that are simultaneously guarded and weakly-acyclic), highlighting the difficulty of reasoning in DatalogMTL$^{\exists}$. Thus, DatalogMTL$^{\exists}$ under natural semantics is undecidable in all case we considered.

Regarding the uniform semantics, we shown that reasoning in DatalogMTL$^{\exists}$ becomes significantly easier. It is still undecidable already for guarded programs, which is again in a significant contrast to Datalog$^{\exists}$. However, in the case of weakly acyclic DatalogMTL$^{\exists}$ programs, both fact entailment and consistency checking are decidable, and in particular, 2-ExpSpace-complete (cf., 2-ExpTime-completeness of weakly acyclic Datalog$^\exists$ [6]). Finally, under CWA, reasoning becomes ExpSpace-complete for full DatalogMTL$^\exists$, as well as for guarded and weakly-acyclic programs.

It is worth emphasising that we use guardedness and weak acyclicity in DatalogMTL$^\exists$ in the same way as they are used in Datalog$^\exists$, so we do not provide any additional conditions for temporal operators. While it may seem naïve to ignore the temporal dimension, our reductions demonstrate essential obstacles in providing time-sensitive versions of these properties. Indeed, our undecidability argument for the guarded fragment shows that we can construct a program simulating computations of a Turing machine, such that the program uses only a single existential rule without temporal operators

$$\exists z Next(y, z) \leftarrow Next(x, y)$$

and the only temporal operator used in the program is $\boxminus_1$, for example, in rules of the form

$$Next(x, y) \leftarrow \boxminus_1 Next(x, y).$$

Hence, extending the guardedness notion to temporal operators in a way that prohibits our reduction would have to restrict the language to the point where even the most basic temporal reasoning is excluded. Notably, our reduction implies also undecidability of existential extensions of much simpler temporal formalisms such as Temporal Datalog [7] or Datalog$_{1S}$ [8] which consider the discrete timeline and allow for very simple temporal operators only.

Similarly, it may seem natural to extend the definition of weak acyclicity by treating each predicate as if it had an additional implicit time attribute, and thus also a corresponding vertex for the time dimension in the dependency graph (cf., [6]). However, our construction in the undecidability proof for weakly acyclic programs under natural semantics would apply even if we use such a time-aware notion of weak acyclicity.

3. Future Research Directions

Among other important fragments for which Datalog$^\exists$ is decidable but which are not mentioned in our analysis so far, are the linear, shy [9], and sticky [10] fragments. Research on such fragments of DatalogMTL$^\exists$ may lead to interesting temporal languages, but it also introduces conceptual challenges.

In particular, the shy fragment identifies (a subset) of programs that are *parsimonious*, which are programs for which inference using the *parsimonious chase* [9] is always correct. The key element of the parsimonious chase is that, roughly speaking, rules are not triggered for an instance when there is a homomorphism of body and head into the instance. While non-existential DatalogMTL admits (transfinite) fixpoint semantics that could be extended to a chase procedure, the temporal dimension of the problem requires a temporal notion of homomorphism. Since every time point effectively has its own interpretation, a temporal notion of homomorphism requires some quantification over time points, e.g., there is some time point t, such that there is a homomorphism into the interpretation for time point t, or for some interval I, there is a homomorphism into every instance in the interval. However, natural definitions of temporal homomorphisms yield highly unintuitive semantics and it therefore remains unclear whether any analogue to the parsimonious chase is possible for DatalogMTL$^\exists$.

The natural immediate questions that are still open concern the decidability of DatalogMTL$^\exists$ for the linear and sticky fragments. However, again techniques from Datalog$^\exists$ are difficult to adapt to the temporal setting and reasoning is significantly more complex even in the linear case. Indeed, fact entailment in linear DatalogMTL is already PSpace-complete in data complexity, i.e., the same complexity as in full DatalogMTL [11]. Understanding the complexity of the linear fragment is also an important precursor to understanding warded DatalogMTL$^\exists$. The warded fragment of Datalog$^\exists$ [12] is known to provide a promising balance of good computational properties and expressivity for real-world applications (see [13]) and may be of interest in the DatalogMTL$^\exists$ context.

Sticky Datalog$^\exists$ enjoys very good computational properties [14], in particular sticky Datalog$^\exists$ programs are FO-rewritable. However, FO-rewritability is clearly not given in the presence of MTL operators. Sticky Datalog$^\exists$ also induces certain beneficial properties in chase procedures and it remains open whether these properties also hold in the temporal setting due to technical differences in the fixpoint semantics between DatalogMTL and Datalog.

Note that many other fragments of Datalog$^\exists$ that have been studied in the literature, such as the weakly-guarded or frontier-guarded fragments (cf., [15, 16]), generalise the guarded fragment and thus can not yield decidable fragments of DatalogMTL$^\exists$.

4. Conclusion & Outlook

The discussion in the previous sections has highlighted that many of the key properties that have been identified for decidable Datalog$^\exists$ fragments are no longer helpful in DatalogMTL$^\exists$. This motivates the study of new restrictions which lead to decidable reasoning in the presence of temporal operators and existential quantifiers. Our results suggest that purely syntactical restrictions are too limited in the presence of temporal operators. In future work we will study the combination of syntactic fragments that control existential quantification, together with properties that are known to control expansion in the time dimension in DatalogMTL, such as MTL-acyclicity [17].

Finally, since existential rules with MTL operators have proven to be useful in various industry applications, such as, technical specifications, verification of banking agreements [18], and fact-checking economic claims [19], we aim to also explore practical algorithms and implementations for reasoning in (fragments and variations of) DatalogMTL$^\exists$.

Acknowledgements

Our work has been supported by the EPSRC projects OASIS (EP/S032347/1), AnaLOG (EP/P025943/1), and UK FIRES (EP/S019111/1), the SIRIUS Centre for Scalable Data Access, and Samsung Research UK. Matthias Lanzinger acknowledges support by the Royal Society "RAISON DATA" project (Reference No. RP\R1\201074).

References

[1] S. Brandt, E. G. Kalaycı, V. Ryzhikov, G. Xiao, M. Zakharyaschev, Querying log data with metric temporal logic, J. Artif. Intell. Res. (2018) 829–877.

[2] R. Koymans, Specifying real-time properties with metric temporal logic, Real-Time Syst. 2 (1990) 255–299.

[3] P. A. Wałęga, B. Cuenca Grau, M. Kaminski, Reasoning over streaming data in metric temporal Datalog, in: Proc. of AAAI, 2019, pp. 1941–1948.

[4] A. Artale, R. Kontchakov, A. Kovtunova, V. Ryzhikov, F. Wolter, M. Zakharyaschev, Ontology-mediated query answering over temporal data: A survey, in: Proc. of TIME, 2017, pp. 1–37.

[5] S. Kikot, V. Ryzhikov, P. A. Wałęga, M. Zakharyaschev, On the data complexity of ontology-mediated queries with MTL operators over timed words, in: Proc. of DL, 2018.

[6] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa, Data exchange: Semantics and query answering, Theor. Comput. Sci. 336 (2005) 89–124. URL: https://doi.org/10.1016/j.tcs.2004.10.033. doi:10.1016/j.tcs.2004.10.033.

[7] A. Ronca, M. Kaminski, B. Cuenca Grau, B. Motik, I. Horrocks, Stream reasoning in temporal Datalog, in: Proc. of AAAI, 2018, pp. 1941–1948.

[8] J. Chomicki, T. Imieliński, Temporal deductive databases and infinite objects, in: Proc. of PODS, 1988, pp. 61–73.

[9] M. Alviano, N. Leone, M. Manna, G. Terracina, P. Veltri, Magic-sets for Datalog with existential quantifiers, in: Proc. of Datalog 2.0, 2012, pp. 31–43. URL: https://doi.org/10.1007/978-3-642-32925-8_5. doi:10.1007/978-3-642-32925-8_5.

[10] A. Calì, G. Gottlob, A. Pieris, Towards more expressive ontology languages: The query answering problem, Artif. Intell. 193 (2012) 87–128. URL: https://doi.org/10.1016/j.artint.2012.08.002. doi:10.1016/j.artint.2012.08.002.

[11] P. A. Wałęga, B. Cuenca Grau, M. Kaminski, E. V. Kostylev, Tractable fragments of Datalog with metric temporal operators, in: Proc. of IJCAI, 2020, pp. 1919–1925.

[12] G. Berger, G. Gottlob, A. Pieris, E. Sallinger, The space-efficient core of Vadalog, in: D. Suciu, S. Skritek, C. Koch (Eds.), Proc. of PODS, ACM, 2019, pp. 270–284. URL: https://doi.org/10.1145/3294052.3319688. doi:10.1145/3294052.3319688.

[13] L. Bellomarini, G. Gottlob, A. Pieris, E. Sallinger, Swift logic for big data and knowledge graphs, in: Proc. of IJCAI, 2017, pp. 2–10. URL: https://doi.org/10.24963/ijcai.2017/1. doi:10.24963/ijcai.2017/1.

[14] A. Calì, G. Gottlob, A. Pieris, Advanced processing for ontological queries, In Proc. of VLDB Endow. 3 (2010) 554–565. URL: http://www.vldb.org/pvldb/vldb2010/pvldb_vol3/R49.pdf. doi:10.14778/1920841.1920912.

[15] G. Gottlob, S. Rudolph, M. Simkus, Expressiveness of guarded existential rule languages, in: R. Hull, M. Grohe (Eds.), Proc. of PODS, ACM, 2014, pp. 27–38. URL: https://doi.org/10.1145/2594538.2594556. doi:10.1145/2594538.2594556.

[16] G. Gottlob, T. Lukasiewicz, A. Pieris, Datalog+/-: Questions and answers, in: Proc. of KR, 2014. URL: http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7965.

[17] P. A. Wałęga, M. Zawidzki, B. Cuenca Grau, Finitely materialisable Datalog programs with metric temporal operators, in: Proc. of KR, 2021, pp. 619–628.

[18] M. Nissl, E. Sallinger, Modelling smart contracts with DatalogMTL, in: Proc. of EDBT/ICDT, 2022.

[19] M. Mori, P. Papotti, L. Bellomarini, O. Giudice, Neural machine translation for fact-checking temporal claims, in: Proc. of FEVER, 2022, pp. 78–82. doi:10.18653/v1/2022.fever-1.8.