On the First-Order Rewritability of Ontology-Mediated Queries in Linear Temporal Logic (Extended Abstract)*

Alessandro Artale[1†], **Roman Kontchakov**[2], **Alisa Kovtunova**[3], **Vladislav Ryzhikov**[2], **Frank Wolter**[4] and **Michael Zakharyaschev**[2]

[1]KRDB Research Centre, Free University of Bozen-Bolzano, Italy

[2]Department of Computer Science and Information Systems, Birkbeck, University of London, UK

[3]Chair for Automata Theory, Technische Universität Dresden, Germany

[4]Department of Computer Science, University of Liverpool, UK

artale@inf.unibz.it, {roman,vlad,michael}@dcs.bbk.ac.uk, alisa.kovtunova@tu-dresden.de, wolter@liverpool.ac.uk

Abstract

We argue that linear temporal logic *LTL* in tandem with monadic first-order logic can be used as a basic language for ontology-based access to temporal data and obtain a classification of the resulting ontology-mediated queries according to the type of standard first-order queries they can be rewritten to.

1 Introduction

Ontology-mediated query (OMQ) answering has recently become one of the most successful applications of description logics (DLs) and semantic technologies. Its main aim is to facilitate user-friendly access to possibly heterogeneous, distributed and incomplete data. To this end, an ontology is employed to provide (*a*) a convenient and uniform vocabulary for formulating queries and (*b*) a conceptual model of the domain for capturing background knowledge and obtaining more complete answers. Thus, instead of querying data directly by means of convoluted database queries, one can use OMQs of the form $(\mathcal{O}, \boldsymbol{q})$, where $\mathcal{O}$ is an ontology and $\boldsymbol{q}$ a query formulated in the vocabulary of $\mathcal{O}$. Under the standard certain answer semantics for OMQs, the answers to $(\mathcal{O}, \boldsymbol{q})$ over a data instance $\mathcal{D}$ are exactly those tuples of individual names from $\mathcal{D}$ that satisfy $\boldsymbol{q}$ in every model of $\mathcal{O}$ and $\mathcal{D}$. Because of this open-world semantics, answering OMQs can be computationally much harder than evaluating standard database queries. For example, answering an atomic query $A(x)$ using an ontology in the standard description logic $\mathcal{ALC}$ can be coNP-hard for data complexity—the complexity measure (adopted in this paper) that regards the OMQ as fixed and the data instance as the only input to the OMQ answering problem. For this reason, weaker description logics (DLs) have been developed to enable not only tractable OMQ answering but even a reduction—known as *first-order* (FO-) *rewritability* [Calvanese *et al.*, 2007b]—of OMQ answering to evaluation of standard relational database queries directly over the data, which is in AC^0 for data complexity. After 15 years of intensive research and coding, ontology-based data access (OBDA, aka virtual knowledge graphs) originally conceived by [Calvanese *et al.*, 2007b; Poggi *et al.*, 2008] has become a well-developed area lying at the crossroads of knowledge representation, databases and computational complexity [Xiao *et al.*, 2018], with multiple real-word applications using mature systems such as Mastro and Ontop; see [Xiao *et al.*, 2019] for a survey.

An important application area where OBDA has so far failed to make a substantial impact is the management of *temporal* data. A typical example is querying data from sensors in complex industrial systems such as gas turbines, where sensors read the temperature, rotor speed, power, etc., in order to analyse and predict their behaviour. One of the problems engineers are facing in this scenario is identifying various events of interest (active power trip, abnormal restart, etc.) based on timestamped sensor readings stored in a database. The OBDA approach could immensely simplify this task for engineers by (i) encoding the definitions of those temporal events in an ontology, thereby making query formulation easier, and by (ii) using background knowledge to compensate for incomplete data due, for example, to sensor failures.

Unfortunately, OBDA languages such as *OWL 2 QL*, standardised by the W3C, provide no means to express temporal events in ontologies because the addition of temporal operators even to inexpressive description logics tends to dramatically increase the complexity of reasoning and, in particular, ruin FO-rewritability of OMQs; see [Lutz *et al.*, 2008; Artale *et al.*, 2017] for surveys. One way to avoid this 'curse of many-dimensional logics' would be to keep *OWL 2 QL* as the ontology language but extend the query language (say, SPARQL) with the relevant temporal operators [Borgwardt *et al.*, 2015]. However, this approach clearly undermines principle (i) of OBDA as the burden of encoding complex temporal events in queries lies solely on the users' shoulders.

Here, we propose an alternative approach to designing logics for temporal OBDA, viz., to start with classical linear temporal logic *LTL* as an ontology language and monadic first-order logic with $<$ over timestamps as a query language. Our first observation is that already this basic temporal OBDA for-

*Extended version published in the AI journal, 2021.

†Contact Author

malism is sufficiently expressive in scenarios where the interaction among the individuals in the object domain—say, between different turbines and their parts—usually captured by binary relations in DLs, is not important and can be neglected.

Example 1 Gas turbines, t, are equipped with multiple sensors, s, measuring the rotor speed, the temperature of the blades, vibration, electric current, etc. Imagine that a relational database in a remote diagnostic centre stores a binary predicate *location*(s, t) saying that sensor s is located in turbine t and a ternary predicate *measurement*(s, v, n) giving the numerical value v of the reading of s at time instant n. The timestamps of sensor readings are synchronised with a central computer clock, and so can be regarded as integers.

When defining important events like active power trip, engineers usually operate with statements such as 'the active power of turbine t measured by s is above 1.5MW at moment n', which can be obtained as database views of the form $\mathit{PowerSensor}^{t,s}_{\geq 1.5}(n)$. We regard these unary predicates as atomic propositions that can be true or false at different moments of time. Omitting t and s to simplify notation, we can then assume that our database $\mathcal{D}$ contains facts of the form

$$\mathit{Pause}(5),\ \mathit{Run}(6),\ \mathit{PowerSensor}_{\geq 1.5}(7),\ \mathit{Run}(8),\ \mathit{Malfunction}(8),\ \mathit{Disabled}(11), \dots$$

based on which we analyse the behaviour of the turbines. As some sensors occasionally fail to send their measurements, we cannot assume the data to be complete. Thus, in our sample data above, the sensor detecting if the turbine is running (by measuring the electric current) failed to send a signal at time instant 7. However, the power sensor attached to the turbine recorded $\geq$ 1.5MW at 7, which should imply that the turbine must have been running at 7. This piece of domain knowledge can be encoded as an ontology axiom

$$\Box(\mathit{PowerSensor}_{\geq 1.5} \to \mathit{Run})$$

with the *LTL*-operator $\Box$ (at all times). Other *LTL* axioms in our example ontology $\mathcal{O}$ could look like

$$\Box(\mathit{Pause} \land \mathit{Run} \to \bot),\ \ \Box(\mathit{Disabled} \to \neg \Diamond_{\scriptscriptstyle F} \mathit{Diagnostics}),$$
$$\Box(\mathit{Malfunction} \to \bigcirc_{\scriptscriptstyle F} \mathit{Pause} \land \Diamond_{\scriptscriptstyle F} \mathit{Diagnostics}).$$

The first of them says that a turbine cannot be paused and running at the same time; the second says that a disabled turbine cannot undergo diagnostics in the future ($\Diamond_{\scriptscriptstyle F}$); and the third axiom asserts that immediately after ($\bigcirc_{\scriptscriptstyle F}$) a malfunction the turbine is paused and will eventually be diagnosed.

Now, if we are interested in maximal intervals of uninterrupted run of a turbine, we can execute the following query formulated in MFO($<$) (monadic first-order logic with $<$):

$$\boldsymbol{q}_1(x, y) \ = \ (y > x) \land \neg \mathit{Run}(x - 1) \land \neg \mathit{Run}(y + 1) \land \forall z \, \big((x \leq z \leq y) \to \mathit{Run}(z)\big),$$

which is mediated by the ontology $\mathcal{O}$ and returns the certain answer $[6, 8]$ over $\mathcal{D}$. The MFO($<$)-query

$$\boldsymbol{q}_2(x, y) = \boldsymbol{q}_1(x, y) \land \exists z \, \big((y < z < y+3) \land \mathit{Diagnostics}(z)\big)$$

finds intervals of maximal continuous run followed by a diagnostics event within 3 time units. The certain answer to the OMQ $(\mathcal{O}, \boldsymbol{q}_2(x, y))$ over $\mathcal{D}$ is $[6, 8]$ as well because $\mathcal{O}$ and $\mathcal{D}$ imply that the diagnostics took place at some (unknown) moment within the interval $[9, 11]$.

2 OMQs in *LTL* and MFO($<$)

Formally, the temporal ontology language we present here operates with *LTL* axioms given in the clausal normal form

$$\Box(C_1 \land \dots \land C_k \ \to \ C_{k+1} \lor \dots \lor C_{k+m}), \qquad (1)$$

where the C_i are *atomic concepts* (unary predicates), possibly prefixed with the temporal operators $\bigcirc_{\scriptscriptstyle F}$ (next time), $\Box_{\scriptscriptstyle F}$ (always in the future) and their past-time counterparts $\bigcirc_{\scriptscriptstyle P}$ and $\Box_{\scriptscriptstyle P}$. An *ontology*, $\mathcal{O}$, is a finite set of such axioms. Note that every finite set of $\Box$-prefixed *LTL*-formulas can be efficiently converted into clausal form without affecting OMQ answering to be defined below. For example, the last axiom of $\mathcal{O}$ from Example 1 can be replaced by $\Box(\mathit{Disabled} \to \Box_{\scriptscriptstyle F} A)$ and $\Box(A \land \mathit{Diagnostics} \to \bot)$ with fresh A.

A *data instance*, $\mathcal{D}$, is a finite set of atoms of the form $A(n)$ with a *timestamp* $n \in \mathbb{Z}$. We query data using MFO($<$) formulas $\boldsymbol{q}(\boldsymbol{x})$ built from unary atoms $A(x)$ and binary atoms $x < y$ as usual in first-order logic. Here, $\boldsymbol{x}$ is the set of free variables in $\boldsymbol{q}$, denoted as *answer* variables. An *ontology-mediated query* is a pair $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$. A *certain answer* to $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$ over $\mathcal{D}$ is any assignment of some timestamps from $\mathcal{D}$ to the answer variables $\boldsymbol{x}$ under which $\boldsymbol{q}$ is true in *all* *LTL*-models of $\mathcal{O}$ and $\mathcal{D}$. Note that we use the standard *strict* semantics of the *LTL* operators (which does not include the current point); see, e.g., [Gabbay *et al.*, 1994; Demri *et al.*, 2016].

We now define the key notion of FO-rewritability. Let $\mathcal{L}$ be a first-order language that is capable of speaking about finite linear orders. An OMQ $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$ is called *$\mathcal{L}$-rewritable* if there is an $\mathcal{L}$-formula $\boldsymbol{Q}(\boldsymbol{x})$ with free variables $\boldsymbol{x}$—an *$\mathcal{L}$-rewriting* of $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$—such that, for any data instance $\mathcal{D}$, a tuple $\boldsymbol{\ell}$ of timestamps from $\mathcal{D}$ is a certain answer to $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$ over $\mathcal{D}$ if and only if $\boldsymbol{Q}(\boldsymbol{\ell})$ is true in $\mathcal{D}$ regarded as an FO-structure. Thus, answering $\mathcal{L}$-rewritable *LTL* OMQs under the open-world semantics as defined above reduces to evaluating their $\mathcal{L}$-rewritings over the original data under the closed-world semantics. In other words, the data complexity of answering those OMQs is the same as the data complexity of evaluating $\mathcal{L}$-formulas. Good target languages $\mathcal{L}$ whose evaluation lies in one of the smallest complexity classes AC^0 are FO($<$) with one built-in predicate $<$ over timestamps and FO($<, \equiv$) that also allows the unary predicates $x \equiv 0 \pmod n$, for any fixed $n > 1$. Executing such $\mathcal{L}$-queries can be done by standard database management systems.

Example 2 (a) The OMQ $(\mathcal{O}, \boldsymbol{q}_1(x, y))$ from Example 1 is FO($<$)-rewritable to the following formula:

$$\boldsymbol{Q}(x, y) = \big[(y > x) \land \pi(x - 1) \land \pi(y + 1) \land \forall z \, \big((x \leq z \leq y) \to \rho(z)\big)\big] \lor \iota,$$

where $\pi(x)$ stands for $\mathit{Pause}(x) \lor \mathit{Malfunction}(x - 1)$, $\rho(x)$ for $\mathit{Run}(x) \lor \mathit{PowerSensor}_{\geq 1.5}(x)$ and ι is a disjunction of sentences such as $\exists z \, (\mathit{Malfunction}(z) \land \mathit{Run}(z + 1))$ and $\exists z \, (\mathit{Pause}(z) \land \mathit{Run}(z))$ expressing inconsistency of $\mathcal{O}, \mathcal{D}$.

(b) Now, suppose an ontology $\mathcal{O}$ contains the axioms

$$\Box(\mathit{Activated} \to \bigcirc^{24}_{\scriptscriptstyle F} \mathit{Diagnostics}),$$
$$\Box(\mathit{Diagnostics} \to \bigcirc^{24}_{\scriptscriptstyle F} \mathit{Diagnostics})$$

saying that the turbine undergoes diagnostics in 24 hours after its activation, and that diagnostics repeats every 24 hours. The OMQ $(\mathcal{O}, \boldsymbol{q}(x))$ with $\boldsymbol{q}(x) = \mathit{Diagnostics}(x)$ is FO$(<, \equiv)$-rewritable to the following formula:

$$\mathit{Diagnostics}(x) \lor \exists y \left[\big(x - y \in 24 + 24\mathbb{N}\big) \land \big(\mathit{Diagnostics}(y) \lor \mathit{Activated}(y)\big) \right],$$

where $x - y \in a + b\mathbb{N}$ with constants a, b—i.e., $x - y = a + bk$, for some $k \in \mathbb{N}$—is expressible via predicates of the form $z \equiv 0 \pmod b$. However, $(\mathcal{O}, \boldsymbol{q})$ is not FO$(<)$-rewritable.

Not all OMQs are FO$(<, \equiv)$-rewritable: e.g., there is an OMQ that checks if the number of some events happened in a given data instance is even or odd, and as well known, the parity function is not in AC^0. This parity OMQ can be rewritten to an FO$(<, \mathrm{MOD})$-formula with quantifiers $\exists^n x\, \varphi(x)$ checking if the number of timestamps ℓ satisfying $\varphi(\ell)$ is divisible by n. Evaluating FO$(<, \mathrm{MOD})$ formulas corresponds to the circuit complexity class ACC^0. Our first result is:

Theorem 3 *Any OMQ with an LTL-ontology and an* MFO$(<)$*-query is* FO(RPR)*- and* MSO$(<)$*-rewritable, and so can be answered in* NC^1 *for data complexity.*

Here, FO(RPR) comprises first-order formulas with relational primitive recursion [Compton and Laflamme, 1990] and MSO$(<)$ monadic second-order formulas with $<$, both of which can be evaluated in NC^1 for data complexity. Note that the SQL:1999 ISO standard contains a WITH RECURSIVE construct, which can represent some FO(RPR)-queries, and that $\mathrm{AC}^0 \subsetneqq \mathrm{ACC}^0 \subseteq \mathrm{NC}^1 \subseteq \textsc{LogSpace} \subseteq \mathrm{P}$. This implies that answering *LTL*/MFO$(<)$ OMQs can be performed by an efficient parallel algorithm; it also means that answering such OMQs can be done using finite automata.

One of the main results of this paper is a uniform syntactical classification of OMQs according to their rewritability type (= data complexity). We classify OMQs $(\mathcal{O}, \boldsymbol{q})$ along three axes: the form of MFO$(<)$-queries $\boldsymbol{q}$, the Boolean shape of the clauses in ontologies $\mathcal{O}$, and the temporal operators that can occur in $\mathcal{O}$.

Before giving a survey of our results in terms of these axes, we discuss our choice of *LTL* and MFO$(<)$ as ontology and query languages. *LTL* can be regarded as a fragment of MFO$(<)$ by spelling out the quantifier patterns defining the temporal operators. This *standard FO-translation* can be done in linear time, and so from a purely technical viewpoint, we can equivalently express any *LTL* ontology in MFO$(<)$. We note, however, that typical ontological axioms are much more intuitive and succinct in *LTL* than in MFO$(<)$. The question then is whether MFO$(<)$ could provide additional expressive power to formulate domain knowledge in the ontology. By a celebrated result in philosophical logic, called Kamp's Theorem [Kamp, 1968], the answer is negative: for every MFO$(<)$-formula φ with one free variable, one can construct an equivalent *LTL* formula φ^* (i.e., $\varphi(\ell)$ is true in a model $\mathcal{I}$ over $\mathbb{Z}$ iff φ^* is true at ℓ in $\mathcal{I}$). It follows, in particular, that axioms $\forall x\, \varphi(x)$ can be translated into *LTL* as $\Box\varphi^*$. In the worst case, this translation is non-elementary, and so there are sentences in MFO$(<)$ that cannot be practically expressed in *LTL*. It seems, however, that such sentences are hardly relevant for the formulation of domain knowledge, and so we focus on *LTL* as our ontology language.

The question whether queries should be formulated in MFO$(<)$ or *LTL* requires a more nuanced answer. First, Kamp's Theorem only holds for formulas with one free variable, and so there are practically relevant MFO$(<)$ queries with more than one answer variable that cannot be expressed in *LTL* (see $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ in Example 1). Second, when formulating queries, one is often interested in non-tree-shaped patterns such as, for instance, the very simple MFO$(<)$-query

$$\boldsymbol{q}(x) = \exists y_1, y_2, z \left[(x < y_1 < z) \land (x < y_2 < z) \land A(y_1) \land B(y_2) \land C(z) \right]$$

whose *LTL*-translation needs all possible linearisations of the variables, and so could be difficult to read. Below, we formulate our results for queries in MFO$(<)$, but also characterise our most important fragment of MFO$(<)$ in terms of *LTL*.

Now, for the first axis, apart from arbitrary MFO$(<)$-queries, we consider *atomic* queries of the form $A(x)$ and *quasi-positive* queries that are built (inductively) from atoms $A(x)$, $(x < x')$, $(x = x')$ and $(x = x' + 1)$ using $\land$, $\lor$, $\forall$, $\exists$ as well as the guarded universal quantification such as

$$\forall y \left((x < y < x') \to \varphi \right)$$

with quasi-positive φ. For example, the standard FO-translation of the *LTL* formula $A \mathsf{U} B$ with the until operator U under the strict semantics is the quasi-positive MFO$(<)$-formula $\exists y \left[(x < y) \land B(y) \land \forall z \left((x < z < y) \to A(z) \right) \right]$. Quasi-positive queries play a fundamental role for the formulation of our results, and so we start by characterising their expressive power. The first characterisation is semantical, in terms of *monotone* MFO$(<)$-formulas, i.e., those that are preserved under any addition of timestamps to the extension of atomic concepts in models. It is not hard to see that all quasi-positive formulas are monotone. The other characterisation is syntactical, in terms of *positive LTL concepts* built from atomic concepts using $\land$, $\lor$ and the temporal operators $\Box_{\!F}$, $\Diamond_{\!F}$, $\bigcirc_{\!F}$, U and their past-time counterparts. It generalises Kamp's Theorem to the class of monotone formulas.

Theorem 4 *An* MFO$(<)$*-formula is monotone iff it is equivalent over* $(\mathbb{Z}, <)$ *to a quasi-positive* MFO$(<)$*-formula.*

(***Monotone Kamp Theorem***) *Every monotone* MFO$(<)$*-formula with one free variable is equivalent over* $(\mathbb{Z}, <)$ *either to a positive LTL concept or to* $\bot$.

The second axis in our classification of OMQs is similar to that for DL ontologies [Calvanese *et al.*, 2007b; Artale *et al.*, 2009]: we distinguish between *Boolean* clauses (1), *Horn* clauses (without $\lor$ on the right-hand side), *Krom* clauses with $k + m \leq 2$, and *core* clauses that are both Horn and Krom.

The third axis is in terms of temporal operators in ontologies. Namely, for any pair $\boldsymbol{c} \in \{\mathit{bool}, \mathit{horn}, \mathit{krom}, \mathit{core}\}$ and $\boldsymbol{o} \in \{\Box, \bigcirc, \Box\bigcirc\}$, we denote by $\mathit{LTL}^{\boldsymbol{o}}_{\boldsymbol{c}}$ the fragment of *LTL* with $\boldsymbol{c}$-clauses (1), in which the temporal concepts C_i may only contain the (future and past) operators indicated in $\boldsymbol{o}$. For example, ontologies given in *Prior-LTL*—one of the first temporal logics going back to Prior [Prior, 1956]—can be converted into $\mathit{LTL}^{\Box}_{\mathit{bool}}$ ontologies. We establish the

following complete uniform syntactical classification of the *LTL*/MFO($<$)-OMQs, complementing Theorem 3:

Theorem 5 *(i) All OMQs with an $\mathit{LTL}^{\Box}_{\mathit{bool}}$ ontology and an atomic query are* FO($<$)*-rewritable.*

(ii) All OMQs with an $\mathit{LTL}^{\Box\bigcirc}_{\mathit{krom}}$ ontology and an atomic query are FO($<, \equiv$)*-rewritable. Some of them are not* FO($<$)*-rewritable.*

(iii) Answering OMQs with an $\mathit{LTL}^{\bigcirc}_{\mathit{horn}}$ ontology and an atomic query is NC^1*-hard.*

(iv) All OMQs with an $\mathit{LTL}^{\Box}_{\mathit{horn}}$ ontology and a quasi-positive MFO($<$) *query are* FO($<$)*-rewritable.*

(v) All OMQs with an $\mathit{LTL}^{\Box\bigcirc}_{\mathit{core}}$ ontology and a quasi-positive MFO($<$) *query are* FO($<, \equiv$)*-rewritable. Some of them are not* FO($<$)*-rewritable.*

(vi) Answering OMQs with an $\mathit{LTL}^{\Box}_{\mathit{krom}}$ ontology and a quasi-positive MFO($<$) *query is* NC^1*-hard.*

3 Outlook

As an alternative to the open-world semantics for OMQ answering considered above, one can use the epistemic semantics (ES) proposed by [Calvanese *et al.*, 2007a]. Under ES, $\neg A(n)$ is regarded to be true if $A(n)$ is not entailed. (The same semantics is used in the standard query language SPARQL for RDF datasets when interpreted with OWL ontologies under the entailment regimes [Glimm and Ogbuji, 2013].) For example, the answer to $\boldsymbol{q}_1$ from Example 1 under ES remains $[6, 8]$. However, $\boldsymbol{q}_2$ has no answers under ES because there is no ℓ in the interval $[9, 11]$ such that *Diagnostics*(ℓ) is entailed. Indeed, the answer $[6, 8]$ to $\boldsymbol{q}_2$ can be deemed meaningless as there is no definite time ℓ for diagnostics. This is one motivation for ES. Another motivation is that, for many OMQ languages, answering FO-queries with $\forall$ and $\neg$ under ES becomes decidable, while it is undecidable under the open-world semantics. As an alternative to our query language, we can use MFO($<$) formulas, in which the atoms are replaced by *LTL*-formulas, under ES. All such OMQs will be FO(RPR)-rewritable. Even lower complexity bounds can be established when *positive LTL*-formulas are used as atoms; see [Artale *et al.*, 2021b].

The classification of *LTL*/MFO($<$) OMQs obtained in this paper is *uniform* in the sense that *all* OMQs in a given syntactical class are $\mathcal{L}$-rewritable, for a target language $\mathcal{L}$, with *some* of them being non-rewritable to any weaker language. Yet, many useful OMQs in a syntactical class can be rewritten to more efficient languages. Ideally, we would like to have an OBDA system that could recognise the optimal rewritability type for any given OMQ—in other words, determine the exact data complexity of answering that OMQ. The same problem has been investigated for datalog queries since the 1980s [Kanellakis, 1990] and for description logic OMQs since the 2010s [Bienvenu *et al.*, 2014; Hernich *et al.*, 2020].

First steps in this direction for OMQs, in which both ontology and query are given in *LTL*, were made in [Ryzhikov *et al.*, 2021]. It was shown that, for any target language $\mathcal{L} \in \{\mathrm{FO}(<), \mathrm{FO}(<, \equiv), \mathrm{FO}(<, \mathrm{MOD})\}$, deciding $\mathcal{L}$-rewritability of OMQs $(\mathcal{O}, \boldsymbol{q})$ with *LTL* queries $\boldsymbol{q}$ is equivalent to deciding $\mathcal{L}$-definability of the regular language given by an exponential-size automaton. The problem of recognising whether a regular language is $\mathcal{L}$-definable has been considered since the 1990s. In particular, it has been shown that deciding FO($<$)-definability of regular languages is PSPACE-complete [Cho and Huynh, 1991] and, recently, that deciding the other types of $\mathcal{L}$-definability is PSPACE-complete, too [Kurucz *et al.*, 2021]. Using these results, [Ryzhikov *et al.*, 2021] prove that deciding $\mathcal{L}$-rewritability of the *LTL* OMQs mentioned above is EXPSPACE-complete and also identify some smaller classes of OMQs in which $\mathcal{L}$-rewritability is decidable in PSPACE (for positive OMQs with a core ontology) and CONP (for atomic OMQs with a Krom ontology). However, the full picture is still far from clear.

LTL OMQs over discrete $(\mathbb{Z}, <)$ are appropriate for modelling synchronous systems with a central clock against which all events can be sequenced. However, in scenarios where sensors report their readings asynchronously, dense $(\mathbb{Q}, <)$ for both timestamps and ontology axioms appears more adequate. A promising temporal KR formalism in this case is metric temporal logic *MTL* that was introduced for modelling and reasoning about real-time systems [Koymans, 1990]. In *MTL*, the temporal operators are indexed by a time interval over which the operator works: for example, $\Diamond_{(0,1.5]} A$ is true at n iff A holds at some n' with $0 < n' - n \leq 1.5$. The interpretation domain is $\mathbb{Q}$ under the continuous semantics and the active domain of the data instance under the pointwise semantics [Ouaknine and Worrell, 2008]. Some use cases, algorithms, and implementations for *MTL* OMQs under the continuous semantics are discussed by [Brandt *et al.*, 2018; Wang *et al.*, 2022]. Finding a syntactical classification of *MTL* OMQs according to their data complexity/rewritability type under the pointwise semantics is substantially more challenging than in the *LTL* case [Ryzhikov *et al.*, 2019]. The relevant complexity classes for answering *MTL* OMQs include (N)LOGSPACE, P and NP, which, apart from the three axes above, also depend on the type of intervals indexing the temporal operators, say semi-infinite (a, ∞), punctual $[a, a]$, or bounded $[0, a)$. The target languages for rewritings include FO(DTC), FO(TC) with (deterministic) transitive closure, and datalog(FO).

If n-ary relations over the object domain need to be taken into account in ontologies and queries, the 1D OMQ languages mentioned above should be extended with appropriate DL or datalog constructs. The FO-rewritability results presented above are lifted, in some cases, to combinations of *LTL* with *DL-Lite* and instance temporal queries in [Artale *et al.*, 2021a]. Other DL examples include atomic queries in the temporal extensions of $\mathcal{EL}$ [Gutiérrez-Basulto *et al.*, 2016]; CQs with guarded negation and metric temporal operators under the minimal-world semantics in the extension of $\mathcal{ELH}$ [Borgwardt *et al.*, 2019]. Extensions of datalog by constraints over a time domain provide an alternative and well investigated approach to querying temporal data [Revesz, 1993; Kanellakis *et al.*, 1995; Toman and Chomicki, 1998].

Acknowledgements

This work was supported by EPSRC UK grants EP/S032207 and EP/S032282.

References

[Artale *et al.*, 2009] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. *J. Artif. Intell. Res.*, 36:1–69, 2009.

[Artale *et al.*, 2017] A. Artale, R. Kontchakov, A. Kovtunova, V. Ryzhikov, F. Wolter, M. Zakharyaschev. Temporal ontology-mediated querying: A survey. In *TIME*, 2017.

[Artale *et al.*, 2021a] A. Artale, R. Kontchakov, A. Kovtunova, V. Ryzhikov, F. Wolter, M. Zakharyaschev. First-order rewritability and complexity of two-dimensional temporal ontology-mediated queries. *CoRR*, abs/2111.06806, 2021.

[Artale *et al.*, 2021b] A. Artale, R. Kontchakov, A. Kovtunova, V. Ryzhikov, F. Wolter, and M. Zakharyaschev. First-order rewritability of ontology-mediated queries in linear temporal logic. *Artif. Intell.*, 299:103536, 2021.

[Bienvenu *et al.*, 2014] M. Bienvenu, B. ten Cate, C. Lutz, and F. Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. on Database Systems*, 39(4):33:1–44, 2014.

[Borgwardt *et al.*, 2015] S. Borgwardt, M. Lippmann, and V. Thost. Temporalizing rewritable query languages over knowledge bases. *J. Web Semant.*, 33:50–70, 2015.

[Borgwardt *et al.*, 2019] S. Borgwardt, W. Forkel, A. Kovtunova. Finding new diamonds: Temporal minimal-world query answering over sparse ABoxes. *RuleML+RR*, 2019.

[Brandt *et al.*, 2018] S. Brandt, E. Kalayci, V. Ryzhikov, G. Xiao, M. Zakharyaschev. Querying log data with metric temporal logic. *J. Artif. Intell. Res.*, 62:829–877, 2018.

[Calvanese *et al.*, 2007a] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. EQL-Lite: Effective first-order query processing in description logics. In *Proc. IJCAI*, pages 274–279, 2007.

[Calvanese *et al.*, 2007b] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reason.*, 39(3):385–429, 2007.

[Cho and Huynh, 1991] S. Cho and D. T. Huynh. Finite-automaton aperiodicity is PSPACE-complete. *Theor. Comp. Sci.*, 88(1):99–116, 1991.

[Compton and Laflamme, 1990] K. J. Compton and C. Laflamme. An algebra and a logic for NC^1. *Inf. Comput.*, 87(1/2):240–262, 1990.

[Demri *et al.*, 2016] S. Demri, V. Goranko, and M. Lange. *Temporal Logics in Computer Science*. CUP, 2016.

[Gabbay *et al.*, 1994] D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*, vol. 1. OUP, 1994.

[Glimm and Ogbuji, 2013] B. Glimm and C Ogbuji. SPARQL 1.1 entailment regimes. W3C recommendation. http://www.w3.org/TR/sparql11-entailment, 2013. Accessed: 2022-03-21.

[Gutiérrez-Basulto *et al.*, 2016] V. Gutiérrez-Basulto, J. C. Jung, and R. Kontchakov. Temporalized EL ontologies for accessing temporal data: Complexity of atomic queries. *Proc. IJCAI*, 2016.

[Hernich *et al.*, 2020] A. Hernich, C. Lutz, F. Papacchini, and F. Wolter. Dichotomies in ontology-mediated querying with the guarded fragment. *ACM Trans. Comput. Log.*, 21(3):20:1–20:47, 2020.

[Kamp, 1968] H. W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, UCLA, 1968.

[Kanellakis *et al.*, 1995] P. Kanellakis, G. Kuper, and P. Revesz. Constraint query languages. *J. Comput. Syst. Sci.*, 51(1):26–52, 1995.

[Kanellakis, 1990] P. C. Kanellakis. Elements of relational database theory. In *Handbook of Theoretical Computer Science*, pp.1073–1156. Elsevier & MIT Press, 1990.

[Koymans, 1990] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[Kurucz *et al.*, 2021] A. Kurucz, V. Ryzhikov, Y. Savateev, and M. Zakharyaschev. Deciding FO-definability of regular languages. In *Proc. RAMiCS*, vol. 13027 of *LNCS*, pages 241–257. Springer, 2021.

[Lutz *et al.*, 2008] C. Lutz, F. Wolter, and M. Zakharyaschev. Temporal description logics: A survey. In *Proc. TIME'08*.

[Ouaknine and Worrell, 2008] J. Ouaknine and J. Worrell. Some recent results in metric temporal logic. In *Proc. FORMATS*, pages 1–13, 2008.

[Poggi *et al.*, 2008] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.

[Prior, 1956] A. Prior. *Time and Modality*. OUP, 1956.

[Revesz, 1993] P. Z. Revesz. A closed-form evaluation for datalog queries with integer (gap)-order constraints. *Theor. Comput. Sci.*, 116(1):117–149, 1993.

[Ryzhikov *et al.*, 2019] V. Ryzhikov, P. A. Walega, and M. Zakharyaschev. Data complexity and rewritability of ontology-mediated queries in metric temporal logic under the event-based semantics. In *Proc. IJCAI*, 2019.

[Ryzhikov *et al.*, 2021] V. Ryzhikov, Y. Savateev, M. Zakharyaschev. Deciding FO-rewritability of ontology-mediated queries in linear temporal logic. In *Proc. TIME*, 2021.

[Toman and Chomicki, 1998] D. Toman and J. Chomicki. Datalog with integer periodicity constraints. *J. Log. Program.*, 35(3):263–290, 1998.

[Wang *et al.*, 2022] D. Wang, P. Hu, P. Walega, B. Cuenca Grau. Meteor: Practical reasoning in datalog with metric temporal operators. In *Proc. AAAI*, 2022.

[Xiao *et al.*, 2018] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev. Ontology-based data access: A survey. In *Proc. IJCAI*, 2018.

[Xiao *et al.*, 2019] G. Xiao, L. Ding, B. Cogrel, and D. Calvanese. Virtual knowledge graphs: An overview of systems and use cases. *Data Intell.*, 1(3):201–223, 2019.