Temporal Ontology-Mediated Queries and First-Order Rewritability: A Short Course

V. Ryzhikov[1], P. A. Wałęga[2], and M. Zakharyaschev[1]

[1] Birkbeck, University of London, U.K.
[2] University of Oxford, U.K.

Abstract. We discuss recent attempts to extend the ontology-based data access (aka virtual knowledge graph) paradigm to the temporal setting. Our main aim is to understand when answering temporal ontology-mediated queries can be reduced to evaluating standard first-order queries over timestamped data and what numeric predicates and operators are required in such reductions. We consider two ways of introducing a temporal dimension in ontologies and queries: using linear temporal logic *LTL* over discrete time and using metric temporal logic *MTL* over dense time.

1 Introduction

Imagine that you are an engineer and that your task is to analyse the behaviour of some complex system (such as gas turbines or drilling rigs) in order to understand what kind of undesirable events have happened in it over the past few days and why. As many others in this weird time of Covid-19 pandemic, you are working from home and can only figure out what has been going on with the system by querying the data, which is automatically collected from the system's sensors and stored in the company's databases. Your intuition and experience suggest that first you should look for *emergency stops* and unusually *high* and *low* values of *temperature*, *rotor speed* and other parameters read by the sensors. Then you would investigate more complex events such as '*purging is over*' that happens, according to the manual, when the main flame was on for the past 10 seconds and also, within the previous 10 minutes, there was a 30-second period when the rotor speed was above 1260 rpm and, at most 2 minutes before the start of that period, the rotor speed was below 1000 rpm for at least 1 minute.

You might be a brilliant engineer with intimate knowledge of your equipment but alas, the odds are you have no clue where the data is stored and in which form. (Your IT expert, Joe, would have told you, if asked politely, that the equipment ID, timestamp and temperature are, respectively, in the first, tenth and fifth columns of a database table `SENSOR_TEM_C`, the rotor speed measurements are in a different table `ROTOR_SP_RPM`, and emergency stops are in a special spreadsheet—but who would you dare ask him in the time of self-isolation with three kids, a wife-mathematician and a grumpy granny in a one-bedroom flat?)

The *ontology-based data access* (OBDA) [73,29,84]—recently rebranded as the *virtual knowledge graph* (VKG) [85]—paradigm has been introduced in the

mid 2000s to facilitate access to data, make it more user-friendly, and remove the dependency on IT experts as far as formalising user queries is concerned. Informally, an OBDA system such as Mastro[3] or Ontop[4] allows the users to think that the data is always at hand in the form of a 'knowledge graph' whose vertices are individuals in the database labelled by classes they belong to and whose directed edges are labelled by relationships between those individuals. For example, the database facts that $t11$ is a gas turbine having rotor $r3$ as its moving part can be thought of as a graph edge labelled by *hasMovingPart* and going from a vertex $t11$ with label *GasTurbine* to a vertex $r3$ with label *Rotor*.

As the names of classes and relationships are familiar to the users (taken from the user manual or a textbook on the relevant domain) and the structure of the data is transparent (directed graph), formulating queries in the OBDA system should be much easier than, say, in a relational database management system (RDBMS)—especially if a visual query interface such as OptiqueVQS [77] is enabled. Moreover, the OBDA system has a secret weapon in the form of an ontology, which is supposed to capture the background knowledge about the domain. In particular, it can contain definitions and descriptions of complex classes and relationships in terms of primitive ones, which can also be utilised in user queries. Thus, the ontology provides the user with a high-level conceptual view of the data and fixes a convenient vocabulary for queries; it supports queries to multiple and possibly heterogeneous data sources, which are of no concern to the user; and it allows the system to enrich incomplete data with background knowledge, which involves reasoning.

Does it sound too good to be true? Let us make the OBDA fairy tale told above more formal and see what it amounts to computationally. Denote by $\mathcal{O}$ the ontology (designed by a domain expert). There exist dozens of ontology languages: description logics (DLs) [12,13] of different expressivity, the Web Ontology Language OWL[5] and its profiles, logic programming and deductive database languages Prolog and datalog, the perennial first-order logic (FO), etc. Let $\mathcal{O}$ be formulated in one of those languages. The knowledge graph, which the user wants to query, is a set, $\mathcal{A}$, of ground atoms of the form $A(a)$ and $P(a, b)$, where A is a unary predicate (class or concept such as *Rotor*) and P a binary predicate (relationship or property such as *hasMovingPart*) in the vocabulary of $\mathcal{O}$. The trouble is that $\mathcal{A}$ does not exist, it is virtual. It can be defined by means of a set, $\mathcal{M}$, of mappings of the form $S(\boldsymbol{x}) \leftarrow \Phi(\boldsymbol{x})$, where S is a unary or binary predicate from $\mathcal{O}$ and Φ a query over one of the datasources, $\mathcal{D}$ (say, an SQL query to a relational database). Thus, $\mathcal{A}$ can be defined as $\mathcal{M}(\mathcal{D})$. The mappings $\mathcal{M}$ are written by an expert with detailed knowledge of both $\mathcal{O}$ and $\mathcal{D}$. The users do not need to know anything about them. The users' only task is to formulate a query $\varphi(\boldsymbol{x})$ over (imaginary) $\mathcal{A}$ in a query language such as SPARQL[6] and press the enter key. The pair $\boldsymbol{q} = (\mathcal{O}, \varphi(\boldsymbol{x}))$ is called an *ontology-mediated query*,

[3] `https://www.obdasystems.com`
[4] `https://ontopic.biz`
[5] `https://www.w3.org/TR/owl2-overview/`
[6] `https://www.w3.org/TR/sparql11-query/`

OMQ for short. The OBDA system is then supposed to find *certain answers to* $\boldsymbol{q}$ over $\mathcal{A}$—actually, over $\mathcal{D}$ as $\mathcal{A}$ does not exist—which are tuples $\boldsymbol{a}$ of individuals from $\mathcal{A}$ such that $\varphi(\boldsymbol{a})$ is a logical consequence of $\mathcal{O}$ and $\mathcal{A}$, in which case we write $\mathcal{O}, \mathcal{A} \models \varphi(\boldsymbol{a})$. Thus, the OBDA system has to somehow compute all tuples $\boldsymbol{a}$ for which $\mathcal{O}, \mathcal{M}(\mathcal{D}) \models \varphi(\boldsymbol{a})$ holds. Whether it is feasible or not depends on the ontology, mapping and query languages involved.

The crucial idea behind the OBDA paradigm is that these languages should be chosen in such a way that the *OMQ answering problem*—'is it the case that $\mathcal{O}, \mathcal{M}(\mathcal{D}) \models \varphi(\boldsymbol{a})$?'—could be reduced to the evaluation problem for standard database queries directly over $\mathcal{D}$. When $\mathcal{D}$ is a relational database, it would be great if the problem of answering $\boldsymbol{q}$ could be somehow reformulated, or *rewritten*, by the OBDA system into an SQL query $\boldsymbol{Q}(\boldsymbol{x})$ with the answer variables $\boldsymbol{x}$, which is then executed over $\mathcal{D}$ by an RDBMS at hand. In our theoretical setting we might as well assume that $\boldsymbol{Q}(\boldsymbol{x})$ is an FO-formula [1] and call an OMQ $\boldsymbol{q} = (\mathcal{O}, \varphi(\boldsymbol{x}))$ *FO-rewritable* if there is an FO-formula $\boldsymbol{Q}(\boldsymbol{x})$, a *rewriting* of $\boldsymbol{q}$, such that, for any data instance $\mathcal{D}$ and any tuple $\boldsymbol{a}$ of individuals in $\mathcal{D}$, we have $\mathcal{O}, \mathcal{M}(\mathcal{D}) \models \varphi(\boldsymbol{a})$ iff $\mathcal{D} \models \boldsymbol{Q}(\boldsymbol{a})$. If the language for mappings $\mathcal{M}$ is sufficiently simple (say, R2RML[7]), then it actually suffices to find an FO-rewriting $\boldsymbol{Q}(\boldsymbol{x})$ of $\boldsymbol{q}$ over possible virtual knowledge graphs $\mathcal{A}$, which can be later transformed to a proper SQL query over $\mathcal{D}$ by the OBDA system using the mappings (for details on this, consult [84] and references therein). Thus, we arrive to the main definition of this paper: $\boldsymbol{Q}(\boldsymbol{x})$ is an *FO-rewriting* of $\boldsymbol{q} = (\mathcal{O}, \varphi(\boldsymbol{x}))$ just in case $\mathcal{O}, \mathcal{A} \models \varphi(\boldsymbol{a})$ iff $\mathcal{A} \models \boldsymbol{Q}(\boldsymbol{a})$, for every VKG $\mathcal{A}$ (in the language of $\mathcal{O}$) and every tuple $\boldsymbol{a}$ of individuals from $\mathcal{A}$.

FO-rewritability is a very strong requirement for ontology and query languages; far from all of them satisfy it. By now, description logics and other fragments of FO that guarantee FO-rewritability are well studied and understood [84]. For example, the *OWL 2 QL* profile of the Web Ontology Language *OWL 2*[8] (underpinned by the *DL-Lite* family of description logics [29,6]) is the W3C standard ontology language for OBDA, which ensures FO-rewritability of all OMQs with an *OWL 2 QL* ontology and a SPARQL or conjunctive query. It is supported by MASTRO and Ontop. So OBDA is not a fairy tale after all.

But can you use it to spot in the sensor data those nasty events that possibly happened in your equipment? Unfortunately, not yet. Because the OBDA framework we discussed above has only been developed and implemented for querying *non-temporal* data using *atemporal* ontologies. Temporal OBDA is still largely work in progress, and the aim of this paper is to present and discuss some of the existing approaches to temporalising ontology-mediated query answering (where the authors have particular research interests). A more comprehensive recent survey of temporal OBDA can be found in [9], though the area is moving forward very fast.

[7] `https://www.w3.org/TR/r2rml/`
[8] https://www.w3.org/TR/owl2-overview/

2 One-Dimensional Temporal OBDA

We begin our discussion of temporal OBDA with a very simple case. Imagine that the system whose behaviour we need to analyse is equipped with a number of sensors, say $S_1, \dots, S_n$. At certain moments of time, t, the system records in a database the current measurement of one or more sensors S_i. So we can assume that the database records take the form $S_i(t, v)$, where t is a moment of time, or a *timestamp*, and v a measurement value. When speaking of temporal events, we often classify those values into qualitative categories, which may vary from one context to another, such as *high temperature* (e.g., $v \geq 300\,\mathrm{C}^\circ$), *low temperature* (say, $v \leq 30\,\mathrm{C}^\circ$), etc. Thus, we may want to think of the record $S_3(t, 350\,\mathrm{C})$ as $\mathit{High}(t)$: at moment t, the temperature was high. The conversion of the original quantitative 2D data $S_i(t, v)$ into qualitative 1D data of the form $A(t)$ can easily be done by mappings, for example,

$$\mathit{High}(t) \leftarrow \texttt{SELECT t FROM } S_3 \texttt{ WHERE v >= 300}.$$

To sum up, let A_i, for $i = 1, 2, \dots$, be a countably infinite list of unary (or monadic) predicate symbols representing such qualitative measurements. Then every data instance we want to query is simply a finite set of atoms of the form $A_i(t)$ with a timestamp t. But what is time?

In general, this is a very difficult question. Remember 'There is a time for everything. [...] Whatever is has already been' (Ecclesiastes) or 'Time is an illusion. Lunchtime doubly so' (Douglas Adams, The Hitchhiker's Guide to the Galaxy)? Even Computer Science uses numerous models of time: linear and branching, discrete and dense, finite and infinite, etc. [36,38,40,34]. So, what can we say about our data instances?

One important property is clear: as we store data about events that have already happened, we may assume that, for any timestamps t and t', either $t < t'$ or $t = t'$, or $t > t'$. In other words, our time is *linear*. Whether the time is discrete or continuous depends on the type of the system we are dealing with. If it is *synchronous* in the sense that all records can only be made at a central clock signal, then time can be thought of as *discrete*, and so the timestamps in each data instance form a finite subset of the natural numbers $\mathbb{N}$ or the integer numbers $\mathbb{Z}$. In this paper, we prefer the integers.

Thus, in the case of discrete time, a *data instance*, $\mathcal{A}$, is a finite set of *ground atoms* of the form $A_i(\ell)$, where $\ell \in \mathbb{Z}$. We denote by $\min \mathcal{A}$ and $\max \mathcal{A}$ the minimal and maximal integer numbers occurring in $\mathcal{A}$. The *active temporal domain* of a data instance $\mathcal{A}$ is the set $\mathsf{tem}(\mathcal{A}) = \{\, n \in \mathbb{Z} \mid \min \mathcal{A} \leq n \leq \max \mathcal{A} \,\}$. To simplify constructions and without much loss of generality, we assume that $\min \mathcal{A} = 0$ and $\max \mathcal{A} \geq 1$.

If our system is *asynchronous*, database records can be made spontaneously, for example, when the current measurement of sensor S_i differs 'substantially' from the previously recorded measurement taken by S_i. In this case, we can model time by the real numbers $\mathbb{R}$ or the rational numbers $\mathbb{Q}$. Since computer words are binary and finite, we prefer to assume that every timestamp is a *dyadic*

rational number of the form $n/2^m$, where $n \in \mathbb{Z}$ and $m \in \mathbb{N}$. The set of these numbers is denoted by $\mathbb{Q}_2$. Note that although rationals such as $1/3$ are not dyadic, by Cantor's theorem, the order $(\mathbb{Q}_2, <)$ is *dense* in the sense that

$$\forall x, y \big((x < y) \to \exists z \, (x < z < y)\big)$$

and isomorphic to the order $(\mathbb{Q}, <)$. Hence, rationals can be approximated with any accuracy by dyadic rationals. In this case, a *data instance* $\mathcal{A}$ is a finite set of ground atoms of the form $A_i(\ell)$ with $\ell \in \mathbb{Q}_2$. The *active temporal domain* of $\mathcal{A}$ is the finite set $\mathsf{tem}(\mathcal{A}) = \{\ell \mid A_i(\ell) \in \mathcal{A}\}$.

Suppose we are interested in finding certain events in a given data instance. An event can be classified as *instantaneous* if it makes sense to say that it happens at a time instant (for example, an emergency stop or a power trip happened at moment t) and *extended* that can happen over a temporal interval (for example, the temperature was rising between t_1 and t_2). In this paper, we mainly consider the former type of events.

A natural language for speaking about temporal instantaneous events over $\mathbb{Z}$ is MFO($<$), *monadic first-order logic* with built-in precedence relation $<$ (cf. [30,52]). More precisely, MFO($<$)-*formulas* are built from unary predicates $A_i(x)$ and binary predicates $x = y$, $x < y$ using the standard Boolean connectives ($\wedge$, $\vee$, $\to$, $\neg$) and first-order quantifiers ($\forall$ and $\exists$). They are interpreted in the usual way in structures, called (*temporal*) *interpretations*, of the form

$$\mathcal{I} \;=\; (\mathbb{Z}, <, A_1^{\mathcal{I}}, A_2^{\mathcal{I}}, \dots),$$

which have domain $(\mathbb{Z}, <)$ and interpret every predicate A_i as a subset $A_i^{\mathcal{I}} \subseteq \mathbb{Z}$. Let $\mathfrak{a}$ be an *assignment* of numbers from $\mathbb{Z}$ to individual variables in MFO($<$)-formulas. Given an MFO($<$)-formula $\varphi(\boldsymbol{x})$ with free variables $\boldsymbol{x} = (x_1, \dots, x_n)$, we write $\mathcal{I} \models \varphi(\mathfrak{a}(\boldsymbol{x}))$ to say that φ is true in $\mathcal{I}$ under the assignment $\mathfrak{a}$ (which replaces $\boldsymbol{x}$ with $\mathfrak{a}(\boldsymbol{x}) = (\mathfrak{a}(x_1), \dots, \mathfrak{a}(x_n))$).

Example 1. We illustrate what can be said in MFO($<$) by a few examples. Suppose $\varphi(t, t') = (t < t') \wedge \neg \exists t'' \, (t < t'' < t')$, where $t < t'' < t'$ abbreviates the formula $(t < t'') \wedge (t'' < t')$. Then, for any interpretation $\mathcal{I}$ and any $m, n \in \mathbb{Z}$, we have $\mathcal{I} \models \varphi(m, n)$ iff $n = m + 1$. It follows that, in MFO($<$)-formulas, we can freely use the 'functions' $t + n$ and $t - n$, for every fixed $n \in \mathbb{Z}$. This allows us to capture events such as *purging is over* from the introduction. For instance, assuming that the clock ticks every second, the following FO-formula $\varphi(t)$ says that the main flame was on for the past 10 seconds and that within the previous 10 minutes, there was a 30-second period when the rotor speed was above 1260 rpm:

$$\begin{aligned}\varphi(t) \;=\; \forall t' \big((t > t' \geq t - 10) \to \mathit{MainFlameOn}(t')\big) \wedge \exists t' \big[(t > t' \geq t - 600) \wedge \\ \forall t'' \big((t' > t'' \geq t' - 30) \to \mathit{RotorSpeedAbove1260}(t'')\big)\big].\end{aligned}$$

We invite the reader to extend this $\varphi(t)$ to a formula $\psi(t)$ that expresses the event *purging is over* described in the introduction. Then the sentence

$$\forall t \, \big[\mathit{PurgingIsOver}(t) \leftrightarrow \psi(t)\big] \tag{1}$$

defines a predicate *PurgingIsOver*(t) that can be used in queries.

By an MFO($<$)-*ontology* we mean any finite set, $\mathcal{O}$, of MFO($<$)-sentences. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{O}$ if $\mathcal{I} \models \varphi$, for every sentence $\varphi \in \mathcal{O}$, in which case we write $\mathcal{I} \models \mathcal{O}$; $\mathcal{I}$ is a *model* of a data instance $\mathcal{A}$ if $\ell \in A^{\mathcal{I}}$ whenever $A(\ell) \in \mathcal{A}$. We write $\mathcal{O} \models \varphi$ to say that $\mathcal{I} \models \varphi$, for every model $\mathcal{I}$ of $\mathcal{O}$.

MFO($<$)-formulas $\psi(\boldsymbol{t})$ with free variables $\boldsymbol{t} = (t_1, \dots, t_m)$ can also be used as queries asking for assignments of timestamps to the *answer variables* $\boldsymbol{t}$ under which the query holds true in relevant interpretations. An *ontology-mediated query* (OMQ) in MFO($<$) is a pair $\boldsymbol{q} = (\mathcal{O}, \psi(\boldsymbol{t}))$, where $\mathcal{O}$ is an ontology and $\psi(\boldsymbol{t})$ a query, both given in MFO($<$). If $\boldsymbol{t}$ is empty ($m = 0$), then $\boldsymbol{q}$ is called a *Boolean* OMQ.

A *certain answer* to an OMQ $\boldsymbol{q} = (\mathcal{O}, \psi(\boldsymbol{t}))$ over a data instance $\mathcal{A}$ is any tuple $\boldsymbol{\ell} = (\ell_1, \dots, \ell_m)$ such that $\ell_i \in \mathsf{tem}(\mathcal{A})$, for $1 \leq i \leq m$, and

$$\mathcal{I} \models \psi(\boldsymbol{\ell}), \qquad \text{for every model } \mathcal{I} \text{ of } (\mathcal{O}, \mathcal{A}). \tag{2}$$

For a Boolean OMQ $\boldsymbol{q}$, a *certain answer* over $\mathcal{A}$ is 'yes' if $\mathcal{I} \models \psi$, for every model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$, and 'no' otherwise. The set of all certain answers to $\boldsymbol{q}$ over $\mathcal{A}$ is denoted by $\mathsf{ans}(\boldsymbol{q}, \mathcal{A})$. As a technical tool in our constructions, we also consider 'certain answers' that range over the whole of $\mathbb{Z}$ rather than only the active temporal domain $\mathsf{tem}(\mathcal{A})$; we denote the set of such certain answers *over* $\mathcal{A}$ *and* $\mathbb{Z}$ by $\mathsf{ans}^{\mathbb{Z}}(\boldsymbol{q}, \mathcal{A})$.

Example 2. Suppose $\mathcal{O} = \{\, \forall t\, (A(t) \to B(t+1)),\ \forall t\, (B(t) \to A(t+1)) \,\}$ and $\mathcal{A} = \{\, A(0), C(1) \,\}$. Then $2n + 1 \in B^{\mathcal{I}}$, for any $n \geq 0$ and any model $\mathcal{I}$ of $(\mathcal{O}, \mathcal{A})$. It follows that, for the OMQ $\boldsymbol{q} = (\mathcal{O}, \exists t' \, ((t' = t + 2) \wedge B(t')))$, we have $\mathsf{ans}^{\mathbb{Z}}(\boldsymbol{q}, \mathcal{A}) = \{\, 2n+1 \mid n \geq -1 \,\}$, while $\mathsf{ans}(\boldsymbol{q}, \mathcal{A}) = \{1\}$ because $\mathsf{tem}(\mathcal{A}) = \{0, 1\}$.

We are now in a position to introduce the central notion of the paper that reduces answering OMQs over data instances $\mathcal{A}$ to evaluation of first-order queries over a finite first-order structure $\mathfrak{S}_{\mathcal{A}}$ with domain $\mathsf{tem}(\mathcal{A})$ ordered by $<$, in which

$$\mathfrak{S}_{\mathcal{A}} \models A(\ell) \quad \text{iff} \quad A(\ell) \in \mathcal{A},$$

for any predicate A and any $\ell \in \mathsf{tem}(\mathcal{A})$.

Definition 3 (FO-rewritability)**.** Let $\mathcal{L}$ be a class of FO-formulas that can be interpreted over structures $\mathfrak{S}_{\mathcal{A}}$. (For example, $\mathcal{L}$ may coincide with FO($<$) or extend it with some standard numeric predicates such as $\textsc{plus}(x, y, z)$, which is true iff $x + y = z$, or even with some operators such as transitive closure or relational primitive recursion.) Let $\boldsymbol{q} = (\mathcal{O}, \psi(\boldsymbol{t}))$ be an OMQ and $\boldsymbol{Q}(\boldsymbol{t})$ a constant-free $\mathcal{L}$-formula with free variables $\boldsymbol{t}$. We call $\boldsymbol{Q}(\boldsymbol{t})$ an $\mathcal{L}$-*rewriting of* $\boldsymbol{q}$ if, for any data instance $\mathcal{A}$, we have $\mathsf{ans}(\boldsymbol{q}, \mathcal{A}) = \{\boldsymbol{\ell} \subseteq \mathsf{tem}(\mathcal{A}) \mid \mathfrak{S}_{\mathcal{A}} \models \boldsymbol{Q}(\boldsymbol{\ell})\}$.[9] We say that $\boldsymbol{q}$ is $\mathcal{L}$-*rewritable* if it has an $\mathcal{L}$-rewriting.

Additionally, as a technical tool to construct $\mathcal{L}$-rewritings, we require the following notion of phantom. Given an OMQ $\boldsymbol{q} = (\mathcal{O}, \psi(t))$ with one answer

[9] Here, we (ab)use set-theoretic notation for lists and write $\boldsymbol{\ell} \subseteq \mathsf{tem}(\mathcal{A})$ to say that every element of $\boldsymbol{\ell}$ is an element of $\mathsf{tem}(\mathcal{A})$.

variable t and any integer number $k \in \mathbb{Z}$, by a *k-phantom* we understand an $\mathcal{L}$-sentence $\Phi^k_{\boldsymbol{q}}$ such that $\mathfrak{S}_{\mathcal{A}} \models \Phi^k_{\boldsymbol{q}}$ iff $\max \mathcal{A} + k \in \mathsf{ans}^{\mathbb{Z}}(\boldsymbol{q}, \mathcal{A})$, for $k > 0$, and $\mathfrak{S}_{\mathcal{A}} \models \Phi^k_{\boldsymbol{q}}$ iff $k \in \mathsf{ans}^{\mathbb{Z}}(\boldsymbol{q}, \mathcal{A})$, for $k < 0$.

We discuss the relevant classes $\mathcal{L}$ of FO-formulas in the Sections 3 and 5.

Remark 4. In the definition above, we did not allow any constants (numbers) from $\mathbb{Z}$ in rewritings. Note, however, that the MFO($<$)-formulas $\neg\exists t' \, (t' < t)$ and $\neg\exists t' \, (t' > t)$ define the minimal and maximal numbers that occur in any given data instance. In view of this, we can use the constants min and max in FO($<$)-rewritings as syntactic sugar.

Example 5. Consider the OMQ $\boldsymbol{q} = (\mathcal{O}, A(t))$, where $\mathcal{O}$ is the same as in Example 2. It is not hard to see that

$$\boldsymbol{Q}(t) \;=\; \exists s \, \big(A(s) \land P(t,s)\big) \;\lor\; \exists s \, \big(B(s) \land Q(t,s)\big)$$

is a rewriting of $\boldsymbol{q}$ in MFO($<$) extended with the predicates $P(t,s)$ saying that $t - s = 2n$ for some $n \in \mathbb{N}$, and $Q(t,s)$ saying that $t - s = 2n + 1$ for some $n \in \mathbb{N}$. Note that the quantified variable s ranges between $\min \mathcal{A}$ and $\max \mathcal{A}$ in any $\mathfrak{S}_{\mathcal{A}}$; in particular, $t \geq s$. Finally, observe that $\boldsymbol{q}$ is *not* FO($<$)-rewritable since properties such as 't is even' are not definable by FO($<$)-formulas [79,64].

FO-rewritability of an OMQ $\boldsymbol{q} = (\mathcal{O}, \psi(\boldsymbol{t}))$ defined above is closely related to the data complexity of the *OMQ answering problem* for $\boldsymbol{q}$: given a data instance $\mathcal{A}$ and a tuple $\boldsymbol{\ell}$ of elements from $\mathsf{tem}(\mathcal{A})$, decide whether $\boldsymbol{\ell} \in \mathsf{ans}(\boldsymbol{q}, \mathcal{A})$. If $\boldsymbol{q}$ is regarded to be fixed and only $\mathcal{A}$ is the input, we speak of the *data complexity* of this problem; if both $\boldsymbol{q}$ and $\mathcal{A}$ are regarded as input, then we speak about the *combined complexity*. For example, evaluating standard FO-queries (without an ontology) over FO-structures—in other words, the *model checking problem*—is PSPACE-complete for combined complexity and in the class AC^0 for data complexity[10] (which is one of the smallest classes in the complexity hierarchy); see, e.g., [64, Chapter 6]. It is to be noted that, to measure the data complexity, we should assume that our data instances $\mathcal{A}$ are encoded as strings that can be given as inputs to computational devices such as Turing machines; see [56,64] for details. Now, if the OMQ $\boldsymbol{q}$ is FO($<$)-rewritable (over data instances represented as such strings), then the OMQ answering problem for $\boldsymbol{q}$ is in LOGTIME-uniform AC^0. In what follows, when discussing various types of FO-rewritablity and the corresponding complexity classes, we do not want to go into the technical details of the string representation of data instances.

For events in *asynchronous* systems, we need interpretations with a dense order (in this paper, we use $\mathbb{Q}_2$), where FO($<$) is not able to express the function

[10] Non-uniform AC^0 is the class of languages computable by bounded-depth polynomial-size circuits with unary NOT-gates and unbounded fan-in AND- and OR-gates. Evaluation of FO($<$)-formulas extended with *arbitrary* numeric predicates is known to be in non-uniform AC^0 for data complexity. If the circuits mentioned above can be generated by a Turing machine in, say, LOGTIME, we speak about LOGTIME-uniform AC^0. For example, evaluation of FO($<$)-formulas extended with PLUS and TIMES is in LOGTIME-uniform AC^0.

$t + a$, for a fixed number $a \in \mathbb{Q}_2$, which is obviously needed in our purging example. To make FO(<) more expressive, we extend it with built-in binary predicates $\delta_{<a}(x, y)$ and $\delta_{=a}(x, y)$, for $a \in \mathbb{Q}_2^+$ (non-negative dyadic numbers), where the former stands for $0 \leq x - y < a$ and the latter for $x - y = a$. This language is denoted by MFO$(<, \delta_{\leq \mathbb{Q}_2}, \delta_{=\mathbb{Q}_2})$ and interpreted in structures of the form

$$\mathcal{I} \;=\; (\mathbb{T}, <, \{\delta_{<a} \mid a \in \mathbb{Q}_2^+\}, \{\delta_{=a} \mid a \in \mathbb{Q}_2^+\}, A_1^{\mathcal{I}}, A_2^{\mathcal{I}}, \dots) \tag{3}$$

with a temporal domain $\mathbb{T} \subseteq \mathbb{Q}_2$. In fact, there are two types of semantics for MFO$(<, \delta_{\leq \mathbb{Q}_2}, \delta_{=\mathbb{Q}_2})$-formulas; cf. [72]. In one of them, known as the *continuous semantics*, the temporal domain of $\mathcal{I}$ is always set to $\mathbb{T} = \mathbb{Q}_2$ in the definition (2) of the set of certain answers $\mathsf{ans}(\boldsymbol{q}, \mathcal{A})$ to an OMQ $\boldsymbol{q}$ over a data instance $\mathcal{A}$. In the alternative *pointwise* (or *event*) *semantics*, for each data instance $\mathcal{A}$, we only consider those models of $(\mathcal{O}, \mathcal{A})$ in (2) that have the domain $\mathbb{T} = \mathsf{tem}(\mathcal{A})$. We illustrate the difference between these two semantics by an example.

Example 6. Consider the OMQ $\boldsymbol{q} = (\mathcal{O}, A(t))$ with

$$\mathcal{O} = \big\{\, \forall t \, \big(\forall t' \, (\delta_{<2}(t, t') \to B(t')) \to B'(t)\big), \\ \forall t \, \big((\exists t' (\delta_{=1}(t, t') \wedge B'(t'))) \to A(t)\big) \,\big\}.$$

Suppose first that $\mathcal{A}_1 = \{B(0), B(1/2), C(3/2)\}$ and the semantics is pointwise. In this case, possible models of $(\mathcal{O}, \mathcal{A}_1)$ have domain $\{0, 1/2, 3/2\}$, and so we obtain $\mathsf{ans}(\boldsymbol{q}, \mathcal{A}_1) = \{3/2\}$ because the first axiom gives B' at 0 and then at $1/2$, from which we derive A at $3/2$ by the second axiom.

Note that $\mathsf{ans}(\boldsymbol{q}, \mathcal{A}_2) = \emptyset$ for $\mathcal{A}_2 = \{B(0), C(3/2)\}$. Under the continuous semantics, we cannot derive B' at 0 and $1/2$ because there exist time instants before 0 and between 0 and $1/2$ where B' can be false. Thus, in this case, $\mathsf{ans}(\boldsymbol{q}, \mathcal{A}_1) = \emptyset$.

3 Ontology-Mediated Queries with *LTL*-Ontologies

In the previous section, we argued that monadic first-order logic MFO(<) provides a natural formalism for ontology-mediated queries in the synchronous case, where the time flow is discrete $(\mathbb{Z}, <)$. Temporal instantaneous events can be described then by MFO(<)-formulas with one free variable. We recall now that, by the celebrated Kamp Theorem [57,74], those formulas have exactly the same expressive power as the formulas of (propositional) *linear temporal logic LTL*, which are built from propositional variables using the Booleans and the temporal operators $\bigcirc_F$ (at the next moment of time), $\Diamond_F$ (eventually), $\Box_F$ (always in the future), $\mathcal{U}$ (until), and their past-time counterparts $\bigcirc_P$ (at the previous moment), $\Diamond_P$ (some time in the past), $\Box_P$ (always in the past) and $\mathcal{S}$ (since); see [68,40,34] and further references therein.

We give the semantics of *LTL*-formulas via their *standard* MFO(<)-*translation* † defined inductively as follows. For an atomic proposition A, we set $A^\dagger = A(t)$.

The translation † commutes with the Booleans in the sense that $(\varkappa_1 \wedge \varkappa_2)^\dagger = \varkappa_1^\dagger \wedge \varkappa_2^\dagger$, $(\neg\varkappa)^\dagger = \neg(\varkappa^\dagger)$, etc. Finally, for the temporal operators, we set:

$$\begin{aligned}
(\bigcirc_{\scriptscriptstyle F}\varkappa)^\dagger &= \varkappa^\dagger\{t+1/t\},\\
(\Box_{\scriptscriptstyle F}\varkappa)^\dagger &= \forall t' \big((t<t') \to \varkappa^\dagger\{t'/t\}\big),\\
(\Diamond_{\scriptscriptstyle F}\varkappa)^\dagger &= \exists t' \big((t<t') \wedge \varkappa^\dagger\{t'/t\}\big),\\
(\varkappa \mathcal{U} \lambda)^\dagger &= \exists t' \big[(t<t') \wedge \lambda^\dagger\{t'/t\} \wedge \forall t'' \big((t<t''<t') \to \varkappa^\dagger\{t''/t\}\big)\big],
\end{aligned}$$

and symmetrically for the 'past' operators, for instance, $(\bigcirc_{\scriptscriptstyle P}\varkappa)^\dagger = \varkappa^\dagger\{t-1/t\}$, $(\Diamond_{\scriptscriptstyle P}\varkappa)^\dagger = \exists t' \big((t>t') \wedge \varkappa^\dagger\{t'/t\}\big)$, etc. In the translation above, t' and t'' are *fresh variables* and $\{t'/t\}$ means a substitution that replaces t with t'. Given a temporal interpretation $\mathcal{I}$ and an *LTL*-formula $\varkappa$, we define the *extension* $\varkappa^\mathcal{I}$ of $\varkappa$ in $\mathcal{I}$ as the set of integers $\varkappa^\mathcal{I} = \{n \in \mathbb{Z} \mid \mathcal{I} \models \varkappa^\dagger(n)\}$.

In this section, we introduce a number of fragments of *LTL* as possible temporal ontology languages and discuss FO-rewritability of various types of OMQs with ontologies in those fragments. Having in mind the OBDA application area for these languages, we somewhat modify the standard *LTL* terminology. For instance, instead of propositional variables, we prefer to speak about atomic concepts (similarly to concept names in Description Logic).

Thus, in this paper, we think of the alphabet of *LTL* as a countably infinite set of *atomic concepts* A_i, for $i < \omega$. *Basic temporal concepts*, C, are defined by the grammar

$$C \quad ::= \quad A_i \quad | \quad \Box_{\scriptscriptstyle F} C \quad | \quad \Box_{\scriptscriptstyle P} C \quad | \quad \bigcirc_{\scriptscriptstyle F} C \quad | \quad \bigcirc_{\scriptscriptstyle P} C. \tag{4}$$

An *LTL-ontology*, $\mathcal{O}$, is a finite set of *clauses* of the form

$$C_1 \wedge \dots \wedge C_k \;\to\; C_{k+1} \vee \dots \vee C_{k+m}, \tag{5}$$

where $k, m \geq 0$ and the C_i are basic temporal concepts. As usual, we denote the empty conjunction ($k = 0$) by $\top$ and the empty disjunction ($m = 0$) by $\bot$. We often refer to the clauses in $\mathcal{O}$ as (*ontology*) *axioms*. Intuitively, the axioms are supposed to hold at *every moment of time*. In other words, (5) is just a shortcut for the MFO(<)-sentence

$$\forall t\, (C_1^\dagger \wedge \dots \wedge C_k^\dagger \;\to\; C_{k+1}^\dagger \vee \dots \vee C_{k+m}^\dagger). \tag{6}$$

It is true in an interpretation $\mathcal{I}$ iff

$$C_1^\mathcal{I} \cap \dots \cap C_k^\mathcal{I} \;\subseteq\; C_{k+1}^\mathcal{I} \cup \dots \cup C_{k+m}^\mathcal{I}.$$

An ontology $\mathcal{O}$ *entails* a clause (5) if this clause is true in every model $\mathcal{I}$ of $\mathcal{O}$.

We classify ontologies by the shape of their axioms[11] and the temporal operators that occur in them. Let $\boldsymbol{c} \in \{\mathit{horn}, \mathit{krom}, \mathit{core}, \mathit{bool}\}$ and $\boldsymbol{o} \in \{\Box, \bigcirc, \Box\bigcirc\}$. By an $\mathit{LTL}_{\boldsymbol{c}}^{\boldsymbol{o}}$*-ontology* we mean any *LTL*-ontology whose clauses satisfy the following restrictions on k and m in (5) indicated by $\boldsymbol{c}$:

[11] This classification originates in the *DL-Lite* family of description logics [6].

***horn*:** $m \leq 1$,
***krom*:** $k + m \leq 2$,
***core*:** $k + m \leq 2$ and $m \leq 1$,
***bool*:** any $k, m \geq 0$,

and may only contain occurrences of the (future and past) temporal operators indicated in $\boldsymbol{o}$ (for example, $\boldsymbol{o} = \Box$ means that only $\Box_F$ and $\Box_P$ may occur in the temporal concepts). Note that an $\mathit{LTL}^{\boldsymbol{o}}_{\boldsymbol{c}}$-ontology of any type may contain *disjointness axioms* of the form $C_1 \land C_2 \to \bot$. Although both $\mathit{LTL}^{\boldsymbol{o}}_{\mathit{krom}}$- and $\mathit{LTL}^{\boldsymbol{o}}_{\mathit{core}}$-ontologies may only have *binary* clauses as axioms (with at most two concepts), only the former are allowed to contain *universal covering axioms* such as $\top \to C_1 \lor C_2$; in other words, $\mathit{core} = \mathit{krom} \cap \mathit{horn}$.

The definition above identifies a seemingly very restricted set of *LTL*-formulas as possible ontology axioms. For example, it completely disallows the use of the standard temporal operators $\Diamond_F$ (sometime in the future), $\Diamond_P$ (sometime in the past), $\mathcal{U}$ (until) and $\mathcal{S}$ (since). Whether or not these operators can be somehow *expressed* in a given fragment $\mathit{LTL}^{\boldsymbol{o}}_{\boldsymbol{c}}$ (in the context of OMQ answering) depends on $\boldsymbol{c}$ and $\boldsymbol{o}$. The following example clarifies the picture.

Example 7. Observe first that the clause $\Diamond_P A \to B$ is equivalent to the $\mathit{LTL}^{\Box}_{\mathit{core}}$ clause $A \to \Box_F B$. Also, the former clause can be expressed in $\mathit{LTL}^{\bigcirc}_{\mathit{core}}$ by three clauses: $A \to \bigcirc_F X$, $X \to \bigcirc_F X$ and $X \to B$, for a fresh atomic concept X that is not supposed to occur in any data instances.

To see why, we need a few definitions. By the *signature* of an ontology we mean the set of atomic concepts that occur in it. An ontology $\mathcal{O}'$ is called a *model conservative extension* of an ontology $\mathcal{O}$ if $\mathcal{O}' \models \mathcal{O}$, the signature of $\mathcal{O}$ is contained in the signature of $\mathcal{O}'$, and every model of $\mathcal{O}$ can be expanded to a model of $\mathcal{O}'$ by providing interpretations of the fresh symbols of $\mathcal{O}'$ and leaving the domain and the interpretation of the symbols in $\mathcal{O}$ unchanged. Observe that if $\boldsymbol{q} = (\mathcal{O}, \varkappa)$ is an OMQ and $\mathcal{O}'$ a model conservative extension of $\mathcal{O}$, then the certain answers to $\boldsymbol{q}$ over a data instance $\mathcal{A}$ in the signature of $\mathcal{O}$ coincide with the certain answers to $\boldsymbol{q}' = (\mathcal{O}', \varkappa)$ over $\mathcal{A}$. Thus, any FO-rewriting of $\boldsymbol{q}'$ is also an FO-rewriting of $\boldsymbol{q}$.

The ontology $\mathcal{O}'$ obtained from $\mathcal{O}$ by replacing $\Diamond_P A \to B$ with $A \to \bigcirc_F X$, $X \to \bigcirc_F X$ and $X \to B$ is a model conservative extension of $\mathcal{O}$, and so we can use it for OBDA in place of $\mathcal{O}$.

The clause $A \to \Diamond_P B$ cannot be expressed in any of the $\mathit{LTL}^{\boldsymbol{o}}_{\mathit{core}}$ fragments but can be expressed in $\mathit{LTL}^{\Box}_{\mathit{krom}}$ by $A \land \Box_P X \to \bot$ and $\top \to X \lor B$. The clauses of the form $A \land B \to C$ are in the $\mathit{LTL}^{\boldsymbol{o}}_{\mathit{horn}}$ fragments but not in $\mathit{LTL}^{\boldsymbol{o}}_{\mathit{core}}$ or $\mathit{LTL}^{\boldsymbol{o}}_{\mathit{krom}}$. The clause $A \mathcal{U} B \to C$ can be expressed in $\mathit{LTL}^{\bigcirc}_{\mathit{horn}}$ by $B \to \bigcirc_P X$ and $X \land A \to \bigcirc_P X$ and $X \to C$, also for a fresh X. The clause $A \to B \mathcal{U} C$ can be expressed in $\mathit{LTL}^{\Box\bigcirc}_{\mathit{bool}}$ using the well-known fixed-point unfolding of $B \mathcal{U} C$ as $\bigcirc_F C \lor (\bigcirc_F B \land \bigcirc_F (B \mathcal{U} C))$, which gives rise to 4 clauses $A \to X$, $X \to \bigcirc_F C \lor \bigcirc_F B$, $X \to \bigcirc_F C \lor \bigcirc_F X$ and $A \to \Diamond_F C$ (the $\Diamond_F$ in the last clause can be replaced with $\Box_F$ as described above).

Exercise 8. Imagine that we would like to query the data about the status of a research article submitted to a certain journal. We are interested in temporal

events such as *Submission*, *Notification*, *Accept*, *Reject*, *Revise*, *Publication*. Our background knowledge about these events can be formulated as an *LTL*-ontology $\mathcal{O}$ with the axioms below. We invite the reader to transform those axioms to the required form (5) using common sense and the previous example.

$$\mathit{Notification} \leftrightarrow \mathit{Reject} \lor \mathit{Accept} \lor \mathit{Revise}, \tag{7}$$

$$\mathit{Reject} \land \mathit{Accept} \to \bot, \quad \mathit{Revise} \land \mathit{Accept} \to \bot, \quad \mathit{Reject} \land \mathit{Revise} \to \bot \tag{8}$$

(at any moment of time, every notification is either a reject, accept, or revision notification, and it can only be one of them)

$$P \to \neg \Diamond_{\scriptscriptstyle P} P \land \neg \Diamond_{\scriptscriptstyle F} P \tag{9}$$

(for every event P we are interested in except *Notification* and *Revise*, P can happen only once for any article)

$$\mathit{Publication} \to \Diamond_{\scriptscriptstyle P} \mathit{Accept}, \quad \mathit{Notification} \to \Diamond_{\scriptscriptstyle P} \mathit{Submission}, \tag{10}$$

$$\mathit{Accept} \to \Diamond_{\scriptscriptstyle F} \mathit{Publication}, \quad \mathit{Submission} \to \Diamond_{\scriptscriptstyle F} \mathit{Notification}, \tag{11}$$

$$\mathit{Revise} \to \Diamond_{\scriptscriptstyle F} \mathit{Notification} \tag{12}$$

(obvious necessary pre-conditions for publication and notification and also the post-conditions—eventual consequences—of acceptance, submission and a revision notification; for simplicity, we assume that after a revision notification the authors always eventually receive a notification regarding a revised version)

$$\mathit{Accept} \lor \mathit{Reject} \to \neg \Diamond_{\scriptscriptstyle F} \mathit{Notification} \tag{13}$$

(acceptance and rejection notifications are final).

We distinguish between three types of ontology-mediated queries (OMQs) with *LTL*-ontologies:

- *Atomic* OMQs (or OMAQs, for short) take the form $\boldsymbol{q} = (\mathcal{O}, A(t))$ with an atomic concept A.
- A *positive ontology-mediated instance query* (OMPIQ) $(\mathcal{O}, \varkappa(t))$ with a *positive LTL*-formula $\varkappa$, which is constructed from atoms in the standard way but using $\land$ and $\lor$ only (or an equivalent MFO(<)-formula with one free variable t, which will be discussed below).
- *Quasi-positive* OMQs (or OMQPQs) $\boldsymbol{q} = (\mathcal{O}, \psi(\boldsymbol{t}))$ have a *quasi-positive* MFO(<)-formula $\psi(\boldsymbol{t})$, which is recursively constructed using $\land$, $\lor$, $\forall$, $\exists$, as well as the *guarded* universal quantification of the form

$$\forall y \, \big((x < y < z) \to \varphi\big), \qquad \forall y \, \big((x < y) \to \varphi\big), \qquad \forall y \, \big((y < z) \to \varphi\big), \tag{14}$$

  where φ is a quasi-positive MFO(<)-formula.
- Finally, general OMQs $\boldsymbol{q} = (\mathcal{O}, \psi(\boldsymbol{t}))$ may have arbitrary MFO(<)-formulas $\psi(\boldsymbol{t})$.

Example 9. Consider $\mathcal{O}$ from Exercise 8. Then $(\mathcal{O}, \mathit{Revise}(t))$ is an OMAQ asking when the paper was sent back for revision. As an example of OMPIQ, consider $(\mathcal{O}, (\Diamond_P \mathit{Revise} \land \mathit{Accept})(t))$ asking when (and whether) the paper was accepted after revision. As an example of OMQPQ, consider $(\mathcal{O}, \psi(t, t'))$, where

$$\psi(t, t') \;=\; \exists x \left((t < x < t') \land \mathit{Revise}(x) \right) \land \mathit{Submission}(t) \land \mathit{Accept}(t'),$$

looking for time intervals $[t, t']$ over which a submitted article underwent at least one revision before acceptance. Finally, an example of a (general) OMQ is $(\mathcal{O}, \psi(t))$ with

$$\psi(t) = \mathit{Submission}(t) \land \neg \exists t' \left((t' > t) \land (\mathit{Accept}(t') \lor \mathit{Reject}(t')) \right).$$

that checks if and when the article was submitted without accept or reject decision so far.

OMAQs are the simplest and arguably most convenient queries from the user's point of view as they presuppose that definitions of relevant events should be provided by the ontology. However, they are not suitable in situations when more than one answer variable is needed as, for example, in the query 'find all timestamps t_1 and t_3 with $t_1 < t_3$, for which there is t_2 such that $t_1 < t_2 < t_3$ and some event (say, the temperature is lower than $50\,\mathrm{C}^\circ$) happens everywhere in the interval $[t_1, t_2)$ and some other event (say, the temperature is higher than $90\,\mathrm{C}^\circ$) happens everywhere in the interval $[t_2, t_3]$'. As we shall see below, arbitrary OMQs have worse computational properties compared to OMQPQs. In fact, one can show the following version of Kamp's Theorem: every consistent quasi-positive MFO(<)-formula $\psi(t)$ with one free variable t is equivalent over $(\mathbb{Z}, <)$ to a positive *LTL*-formula constructed using any temporal operators and the 'positive' Boolean connectives $\land$ and $\lor$ only [10]. Another important semantic characterisation of quasi-positive formulas is as follows. Given temporal interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$, we write $\mathcal{I}_1 \preceq \mathcal{I}_2$ if $A^{\mathcal{I}_1} \subseteq A^{\mathcal{I}_2}$, for every atomic concept A. An MFO(<)-formula $\psi(\boldsymbol{t})$ is called *monotone* if $\mathcal{I}_1 \models \psi(\boldsymbol{n})$ and $\mathcal{I}_1 \preceq \mathcal{I}_2$ imply $\mathcal{I}_2 \models \psi(\boldsymbol{n})$, for any tuple $\boldsymbol{n}$ in $\mathbb{Z}$. Now, one can show [10] that an MFO(<)-formula is monotone iff it is equivalent over $(\mathbb{Z}, <)$ to a quasi-positive MFO(<)-formula.

Let us now focus on rewritability of *LTL* OMQs. Is there a 'standard' query language into which any such OMQ can be rewritten? The following example shows that FO(<), even extended with arbitrary arithmetic predicates, is not powerful enough for this purpose:

Example 10. Consider the OMAQ $\boldsymbol{q} = (\mathcal{O}, B_0)$, where $\mathcal{O}$ consists of the axioms

$$\bigcirc_P B_k \land A_0 \to B_k \quad \text{and} \quad \bigcirc_P B_{1-k} \land A_1 \to B_k, \quad \text{for} \;\; k = 0, 1.$$

For every binary word $\boldsymbol{e} = e_1 \dots e_n \in \{0,1\}^n$, we take the data instance $\mathcal{A}_{\boldsymbol{e}} = \{ B_0(0) \} \cup \{ A_{e_i}(i) \mid 0 < i \leq n \}$. It is not hard to check that n is a certain answer to $\boldsymbol{q}$ over $\mathcal{A}_{\boldsymbol{e}}$ iff the number of 1s in $\boldsymbol{e}$ is even (PARITY): intuitively, the word is processed starting from the minimal timestamp and moving towards

the maximal one, and the first axiom preserves B_i if the current symbol is 0, whereas the second axiom toggles B_i if the current symbol is 1. As PARITY is not in AC^0 [39], it follows that $\boldsymbol{q}$ is not FO-rewritable even if *arbitrary* numeric predicates are allowed in rewritings.

A natural and more expressive target language for rewritings is FO(RPR) that extends FO with the successor relation and *relational primitive recursion* (RPR, for short). Evaluation of FO(RPR)-formulas is known to be NC^1-complete[12] for data complexity [32], with $\mathrm{AC}^0 \subsetneq \mathrm{NC}^1 \subseteq \mathrm{L}$. We remind the reader that, using RPR, we can construct formulas such as

$$\Phi \;=\; \left[\begin{array}{l} Q_1(\boldsymbol{z}_1, t) \equiv \Theta_1\big(\boldsymbol{z}_1, t, Q_1(\boldsymbol{z}_1, t-1), \dots, Q_n(\boldsymbol{z}_n, t-1)\big) \\ \dots \\ Q_n(\boldsymbol{z}_n, t) \equiv \Theta_n\big(\boldsymbol{z}_n, t, Q_1(\boldsymbol{z}_1, t-1), \dots, Q_n(\boldsymbol{z}_n, t-1)\big) \end{array}\right] \Psi,$$

where the part of Φ within $[\dots]$ defines recursively, via the FO(RPR)-formulas Θ_i, the interpretations of the predicates Q_i in the FO(RPR)-formula Ψ (see Example 11 below). Note that the recursion starts at $t = 0$ and assumes that $Q_i(z, -1)$ is false for all Q_i, $i = 1, \dots, n$, and all z. Thus, the truth value of $Q_i(z, 0)$ is computed by substituting falsehood $\bot$ for all $Q_i(z, -1)$. For every $t = 1, 2, \dots$, the recursion is then applied in the obvious way.

We illustrate relational primitive recursion by a concrete example.

Example 11. The OMQ $\boldsymbol{q} = (\mathcal{O}, B_0)$ from Example 10 can be rewritten to the following FO(RPR)-formula:

$$\boldsymbol{Q}(t) \;=\; \left[\begin{array}{l} Q_0(t) \equiv \Theta_0 \\ Q_1(t) \equiv \Theta_1 \end{array}\right] Q_0(t),$$

where, for $k = 0, 1$,

$$\Theta_k(t, Q_0(t-1), Q_1(t-1)) \;=\; B_k(t) \;\vee\; \big(Q_k(t-1) \wedge A_0(t)\big) \;\vee\; \big(Q_{1-k}(t-1) \wedge A_1(t)\big).$$

As noted above, the recursion starts from the minimal timestamp 0 in the data instance (with $Q_i(-1)$ regarded as false) and proceeds to the maximal one.

The next theorem shows that all *LTL* OMQs can be rewritten into FO(RPR). As follows from [32, Proposition 4.3], this means that we can also rewrite our OMQs into the language MSO(<) of *monadic second-order* formulas that are built from atoms of the form $A(t)$ and $t < t'$ using the Booleans, first-order quantifiers $\forall t$ and $\exists t$, and second-order quantifiers $\forall A$ and $\exists A$ [28].

Remark 12. It is worth reminding the reader (see [79,78,33] for details) that, by the Büchi–Elgot–Trakhtenbrot Theorem [28,35,81], MSO(<)-sentences define exactly the class of regular languages. FO(RPR), extended with the predicates PLUS and TIMES or, equivalently, with one predicate BIT [56], captures exactly the languages in NC^1 (which are not necessarily regular) [32].

[12] NC^1 is the class of languages computable by a family of polynomial-size logarithmic-depth circuits with gates of at most two inputs.

Theorem 13. *Every LTL OMQ is* FO(RPR)- *and* MSO(<)*-rewritable, and so can be answered in* NC^1 *for data complexity.*

Note that the SQL:1999 ISO standard contains a WITH RECURSIVE construct that allows users to implement various FO-queries with relational primitive recursion such as the query in Example 11, which cannot be expressed in FO without recursion.

The next example shows that our OMQs can simulate arbitrary finite automata, and so FO(RPR) appears to be an optimal target language for rewritings in general (since there exist NC^1-complete regular languages).

Example 14. Let $\mathfrak{A}$ be a DFA with a tape alphabet Γ, a set of states Q, an initial state $q_0 \in Q$, an accepting state $q_1 \in Q$ and a transition function $\to$: we write $q \to_e q'$ if $\mathfrak{A}$ moves to a state $q' \in Q$ from a state $q \in Q$ while reading $e \in \Gamma$. (Without loss of generality we assume that $\mathfrak{A}$ has only one accepting state.) We take atomic concepts A_e for tape symbols $e \in \Gamma$ and atomic concepts B_q for states $q \in Q$, and consider the OMAQ $\boldsymbol{q} = (\mathcal{O}, B_{q_0})$, where

$$\mathcal{O} \;=\; \big\{\, \bigcirc_{\!P} B_{q'} \land A_e \to B_q \mid q \to_e q' \,\big\}.$$

For any input word $\boldsymbol{e} = (e_1 \dots e_n) \in \Gamma^*$, we set

$$\mathcal{A}_{\boldsymbol{e}} \;=\; \big\{\, B_{q_0}(0) \,\big\} \;\cup\; \big\{\, A_{e_i}(i) \mid 0 < i \le n \,\big\}.$$

It is easy to see that $\mathfrak{A}$ accepts $\boldsymbol{e}$ iff $\max(\mathcal{A}_{\boldsymbol{e}}) \in \mathsf{ans}(\boldsymbol{q}, \mathcal{A}_{\boldsymbol{e}})$. We invite the reader to show that $\mathfrak{A}$ accepts $\boldsymbol{e}$ iff $0 \in \mathsf{ans}(\boldsymbol{q}', \mathcal{A}_{\boldsymbol{e}})$, where $\boldsymbol{q}' = (\mathcal{O}', \psi(t))$,

$$\mathcal{O}' \;=\; \big\{\, \overline{B}_q \land B_q \to \bot,\ \top \to B_q \lor \overline{B}_q \;\mid\; q \in Q \,\big\},$$

and $\psi(t)$ is the standard FO-translation of the *LTL*-formula

$$\varkappa \;=\; B_{q_1} \lor \bigvee_{q \to_e q'} \Diamond_{\!P}^{+} (\bigcirc_{\!P} B_{q'} \land A_e \land \overline{B}_q),$$

where $\Diamond_{\!P}^{+} C$ is an abbreviation for $C \lor \Diamond_{\!P} C$. (Hint: $\overline{B}_q$ represents the complement of B_q, and $\varkappa$ is equivalent to formula $\big[\bigwedge_{q \to_e q'} \Box_{\!P}^{+} (\bigcirc_{\!F} B_{q'} \land A_e \to B_q)\big] \to B_{q_1}$, where $\Box_{\!P}^{+} C$ is an abbreviation for $C \land \Box_{\!P} C$.)

Now we present classes of OMQs that are rewritable to FO-formulas without recursion. Namely, we have rewritings of two types. One type is the usual FO(<)-formulas. The other one comprises $\mathrm{FO}(<, \equiv_{\mathbb{N}})$-formulas that can use numeric predicates $x \equiv 0 \,(\mathrm{mod}\ n)$, for any fixed $n \in \mathbb{N}$. (As shown in [16], $\mathrm{FO}(<, \equiv_{\mathbb{N}})$ is exactly the class of regular languages in AC^0.)

Exercise 15. Note that FO(RPR) does not contain $x < y$ or $x \equiv 0 \,(\mathrm{mod}\ n)$, for $n > 1$, as atoms. We invite the reader to show that both of them can be expressed in FO(RPR) in the sense that (i) there exists an FO(RPR)-formula $\varphi_<(x, y)$ such that $\mathfrak{S} \models \varphi_<(a, b)$ iff $\mathfrak{S} \models a < b$, for any FO-structure $\mathfrak{S}$ and a, b in its domain; and (ii) for any $n > 1$, there exists an FO(RPR)-formula

$\varphi_n(x)$ such that $\mathfrak{S} \models \varphi_n(a)$ iff $\mathfrak{S} \models a \equiv 0 \,(\mathrm{mod}\, n)$, for any FO-structure $\mathfrak{S}$ and a in its domain. (A solution to (i) can be found in [32] and a solution to (ii) in [10].) As a consequence, we obtain that, for every FO$(<, \equiv_{\mathbb{N}})$-formula, there is an equivalent FO(RPR)-formula.

Theorem 16. *Any* $\mathit{LTL}^{\bigcirc}_{krom}$ *OMAQ is* FO$(<, \equiv_{\mathbb{N}})$*-rewritable, and so can be answered in* AC^0 *for data complexity.*

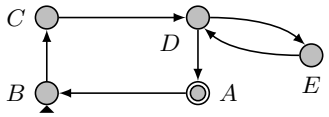
Proof. (Sketch) Suppose $\boldsymbol{q} = (\mathcal{O}, A)$ is an $\mathit{LTL}^{\bigcirc}_{krom}$ OMAQ. By a *literal*, L, we mean an atomic concept in $\boldsymbol{q}$ or its negation. We use $\bigcirc^n L$ in place of $\bigcirc^n_F L$ if $n > 0$, L if $n = 0$, and $\bigcirc^{-n}_P L$ if $n < 0$. We write $\mathcal{O} \models L \to \bigcirc^k L'$ if $\mathcal{I} \models \forall t\, (L \to \bigcirc^k L')^\dagger$ in every model $\mathcal{I}$ of $\mathcal{O}$. For any data instance $\mathcal{A}$ consistent with $\mathcal{O}$, we have:

$$\ell \in \mathsf{ans}^{\mathbb{Z}}(\boldsymbol{q}, \mathcal{A}) \quad \text{iff} \quad \text{either } \mathcal{O} \models \top \to A \\ \text{or } \mathcal{O} \models B \to \bigcirc^{\ell - n} A, \text{ for some } B(n) \in \mathcal{A}.$$

Given literals L and L', let $\mathfrak{A}_{L,L'}$ be an NFA whose tape alphabet is $\{0\}$, the states are the literals, with L initial and L' accepting, and whose transitions are of the form $L_1 \to_0 L_2$, for $\mathcal{O} \models L_1 \to \bigcirc L_2$ (without loss of generality we assume that $\mathcal{O}$ does not contain nested $\bigcirc$). It is easy to see that $\mathfrak{A}_{L,L'}$ accepts 0^k $(k > 0)$ iff $\mathcal{O} \models L \to \bigcirc^k L'$. By [31,80], there are $N = O(|\mathfrak{A}_{L,L'}|^2)$ arithmetic progressions $a_i + b_i\mathbb{N} = \{a_i + b_i \cdot m \mid m \geq 0\}$, $1 \leq i \leq N$, such that $0 \leq a_i, b_i \leq |\mathfrak{A}_{L,L'}|$ and $\mathfrak{A}_{L,L'}$ accepts 0^k iff $k \in a_i + b_i\mathbb{N}$ for some i, $1 \leq i \leq N$. These progressions give rise to the FO-rewriting we need. To illustrate, suppose $\boldsymbol{q} = (\mathcal{O}, A)$ and

$$\mathcal{O} = \{A \to \bigcirc B,\ B \to \bigcirc C,\ C \to \bigcirc D,\ D \to \bigcirc A,\ D \to \bigcirc E,\ E \to \bigcirc D\}.$$

The NFA $\mathfrak{A}_{B,A}$ (more precisely, the states reachable from B) is shown below,



and for $L \in \{A, C, D, E\}$, $\mathfrak{A}_{L,A}$ is the same NFA but with the initial state L. It is readily seen that $\mathfrak{A}_{B,A}$ accepts 0^k iff $k \in 3 + 2\mathbb{N}$, which can be described by the formula

$$\varphi_{B,A}(t) = \exists s \left(B(s) \ \wedge \ \big(t - s \in a + b\mathbb{N}\big)\right),$$

where $a = 3$, $b = 2$, and $\big(t - s \in 3 + 2\mathbb{N}\big)$ is an abbreviation for

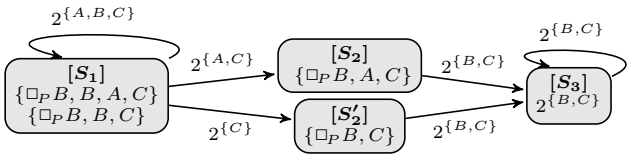
$$\exists n \,[(n = s + 3) \wedge \big(((n \equiv 0 \,(\mathrm{mod}\, 2)) \wedge (t \equiv 0 \,(\mathrm{mod}\, 2))) \vee \\ \exists n', t' \,((n' = n + 1) \wedge (t' = t + 1) \wedge (n' \equiv 0 \,(\mathrm{mod}\, 2)) \wedge (t' \equiv 0 \,(\mathrm{mod}\, 2)))\big)]$$

(the reader is invited to provide the definition of $\big(t - s \in a + b\mathbb{N}\big)$ for arbitrary a and b). Similarly, for $\mathfrak{A}_{E,A}$, we have $a = b = 2$. (Note that in general more than one progression is needed to characterise automata $\mathfrak{A}_{L,A}$.) To obtain an FO$(<, \equiv_{\mathbb{N}})$-rewriting of $\boldsymbol{q}$, we take a disjunction of $\varphi_{L,A}(t)$, for all literals L. One can also construct any phantom $\Phi^k_{(\mathcal{O},A)}$ in FO$(<, \equiv_{\mathbb{N}})$. ❑

Theorem 17. *Any $LTL^{\Box}_{bool}$ OMAQ is* FO(<)*-rewritable, and so can be answered in* AC^0 *for data complexity.*

Proof. (Sketch) Given an $LTL^{\Box}_{bool}$-ontology $\mathcal{O}$, we construct an NFA $\mathfrak{A}$ that takes as input a data instance $\mathcal{A}$ written as the word $\mathcal{A}_0, \dots, \mathcal{A}_k$, where $k = \max \mathcal{A}$ and $\mathcal{A}_i = \{A \mid A(i) \in \mathcal{A}\}$. Let Σ be the set of temporal concepts in $\mathcal{O}$ and their negations. Each state of $\mathfrak{A}$ is a maximal set $S \subseteq \Sigma$ that is consistent with $\mathcal{O}$; let $\boldsymbol{S}$ be the set of all such states. For $S, S' \in \boldsymbol{S}$ and a tape symbol (set of concept names) X, we set $S \to_X S'$ just in case $X \subseteq S'$, $\Box_F \beta \in S$ iff $\beta, \Box_F \beta \in S'$, and $\Box_P \beta \in S'$ iff $\beta, \Box_P \beta \in S$. A state $S \in \boldsymbol{S}$ is *accepting* if $\mathfrak{A}$ has an infinite 'ascending' chain $S \to_\emptyset S_1 \to_\emptyset \dots$; S is *initial* if $\mathfrak{A}$ has an infinite 'descending' chain $\cdots \to_\emptyset S_1 \to_\emptyset S$. The NFA $\mathfrak{A}$ *simulates* $\mathcal{O}$ in the following sense: for any ABox $\mathcal{A}$, concept name A and $\ell \in \mathbb{Z}$, we have $\ell \in \mathsf{ans}^{\mathbb{Z}}((\mathcal{O}, A), \mathcal{A})$ iff $\mathfrak{A}$ does not contain an *accepting path* $S_0 \to_{X_1} \cdots \to_{X_m} S_m$ (S_0 initial and S_m accepting) such that $A \notin S_\ell$, $X_{i+j} = \mathcal{A}_j$ if $0 \le j \le k$, and $X_j = \emptyset$ otherwise, for some i, $0 < i \le m - k$.

Define an equivalence relation, $\sim$, on $\boldsymbol{S}$ by taking $S \sim S'$ iff $S = S'$ or $\mathfrak{A}$ has a cycle with both S and S'. Let $[S]$ be the $\sim$-equivalence class of S. One can check that $S \to_X S'$ implies $S_1 \to_X S'$, for any $S_1 \in [S]$. Let $\mathfrak{A}'$ be the NFA with states $[S]$, for $S \in \boldsymbol{S}$, and transitions $[S] \to_X [S']$ iff $S_1 \to_X S'_1$, for some $S_1 \in [S]$ and $S'_1 \in [S']$. The initial (accepting) states of $\mathfrak{A}'$ are all $[S]$ with initial (accepting) S. The NFA $\mathfrak{A}'$ also simulates $\mathcal{O}$ and contains no cycles other than trivial loops, which makes it possible to express the simulation condition by an FO(<)-formula. For example, $\mathfrak{A}'$ for $\mathcal{O} = \{A \to \Box_P B,\ \Box_P B \to C\}$ is shown below, where all states are initial and accepting, and negated concepts omitted:



Let $\boldsymbol{q} = (\mathcal{O}, C)$. Take all accepting paths π in $\mathfrak{A}'$ with pairwise distinct states at least one of which has a set without C. Thus, for $\pi = [S_1] \to_{\{A\}} [S_2] \to_\emptyset [S_3]$, a set in $[S_3]$ has no C, and the simulation condition for π, which makes sure that $\neg C$ holds at t, can be written as

$$\begin{aligned}\exists t_1, t_2 \big[\forall t' \big((t' < t_1) \to \mathsf{loop}_{[S_1]}(t')\big) \wedge \mathsf{sym}_{\{A\}}(t_1) \wedge \\ \forall t' \left((t_1 < t' < t_2) \to \mathsf{loop}_{[S_2]}(t')\right) \wedge \mathsf{sym}_\emptyset(t_2) \wedge \\ \forall t' \left((t' > t_2) \to \mathsf{loop}_{[S_3]}(t')\right) \wedge (t \ge t_2) \wedge \neg C(t) \big],\end{aligned}$$

where $\mathsf{sym}_{\{A\}}(t) = A(t) \wedge \neg B(t) \wedge \neg C(t)$ and $\mathsf{sym}_\emptyset(t) = \neg A(t) \wedge \neg B(t) \wedge \neg C(t)$ define transitions $\to_{\{A\}}$ and $\to_\emptyset$ in π, and $\mathsf{loop}_{[S_1]} = \top$, $\mathsf{loop}_{[S_3]} = \neg A(t)$ and $\mathsf{loop}_{[S_2]} = \bot$ say that $[S_1]$ and $[S_3]$ have loop transitions with any input and any input but A, respectively, but $[S_2]$ has no loop. To obtain an FO(<)-rewriting of $\boldsymbol{q}$, we take a disjunction of such formulas for all accepting paths π in $\mathfrak{A}'$ and negate it. It is also possible to construct any phantom $\Phi^k_{(\mathcal{O},A)}$ in FO(<). ❑

Theorem 18. *Any* $LTL^{\Box\bigcirc}_{krom}$ *OMAQ is* $\mathrm{FO}(<,\equiv_{\mathbb{N}})$*-rewritable, and so can be answered in* AC^0 *for data complexity.*

Proof. (Sketch) The proof utilises the monotonicity of the $\Box$ operators, similarly to the proof of Theorem 17. However, the latter relies on partially-ordered NFAs accepting the models of $(\mathcal{O},\mathcal{A})$, which do not work in the presence of $\bigcirc$. Our key observation here is that every model of $(\mathcal{O},\mathcal{A})$ has at most $O(|\mathcal{O}|)$ timestamps such that the same $\Box$-concepts hold between any two nearest of them. The placement of these timestamps and their concept-types can be described by an $\mathrm{FO}(<)$-formula. However, to check whether these types are compatible (i.e., satisfiable in some model), we require $\mathrm{FO}(<,\equiv_{\mathbb{N}})$-formulas similar to those in the proof of Theorem 16. ❑

So far, we have considered the simplest class of OMQs with atomic queries. But what if we are interested in spotting complex events whose definitions are not provided by the available ontology? Or what if we need queries with multiple answer variables? Can we still rewrite some of the resulting OMQs into FO-queries without recursion? First, observe that we cannot use arbitrary MFO-formulas as queries if we want to achieve FO-rewritability without recursion. This is the case already for OMQs with empty ontologies as shown by Example 14. It also follows from the same example that OMQs $\boldsymbol{q} = (\mathcal{O}, \psi(\boldsymbol{t}))$ with monotone (quasi-positive) ψ and $\mathcal{O}$ containing only binary disjunctions, can simulate an arbitrary DFA.

On the other hand, we show now how to obtain a strong positive rewritability result for OMQs with a Horn ontology and a monotone query. The key property of Horn ontologies required in the proof of this result is the following fact, which is well known and documented in the Prolog and datalog settings [4,1]. The *intersection* of two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ is the interpretation $\mathcal{I}_3$ such that $A^{\mathcal{I}_3} = A^{\mathcal{I}_1} \cap A^{\mathcal{I}_2}$, for every atomic concept A.

Example 19. Consider $\mathcal{I}_1$ with $A^{\mathcal{I}_1} = \{0, 5\}$, $B^{\mathcal{I}_1} = \{n \in \mathbb{Z} \mid n > 0\}$ and $\mathcal{I}_2$ with $A^{\mathcal{I}_2} = \{-5, 0\}$, $B^{\mathcal{I}_2} = \mathbb{Z}$. Then the intersection of $\mathcal{I}_1$ and $\mathcal{I}_2$ is $\mathcal{I}_3$ with $A^{\mathcal{I}_3} = \{0\}$, $B^{\mathcal{I}_3} = \{n \in \mathbb{Z} \mid n > 0\}$.

In general, if we take a pair of models of $(\mathcal{O},\mathcal{A})$, their intersection does not need to be a model of $(\mathcal{O},\mathcal{A})$. The reader is invited to verify this on the simple KB $(\{A \to B \lor C\}, \{A(0)\})$. However, for $\mathcal{O}$ in $LTL^{\Box\bigcirc}_{horn}$, the intersection of models of $(\mathcal{O},\mathcal{A})$ will be always a model of $(\mathcal{O},\mathcal{A})$, too. Indeed, in Example 19 both $\mathcal{I}_1$ and $\mathcal{I}_2$ are models of $(\mathcal{O},\mathcal{A}) = (\{A \to \Box_F B\}, \{A(0)\})$ and so is their intersection $\mathcal{I}_3$. The *canonical* (or *minimal*) *model* $\mathcal{C}_{\mathcal{O},\mathcal{A}}$ of $(\mathcal{O},\mathcal{A})$, is the intersection of all the models of $(\mathcal{O},\mathcal{A})$. For $(\{A \to \Box_F B\}, \{A(0)\})$, the canonical model is $\mathcal{I}_3$ from Example 19.

Theorem 20. *Let* $\boldsymbol{q} = (\mathcal{O}, \psi(\boldsymbol{t}))$ *be an OMQPQ with an* $LTL^{\Box\bigcirc}_{horn}$*-ontology* $\mathcal{O}$ *and let* $\mathcal{L}$ *be* $\mathrm{FO}(<)$ *or* $\mathrm{FO}(<,\equiv_{\mathbb{N}})$*. Suppose that, for each atomic* A *in* ψ*,*

- *every OMAQ* $(\mathcal{O}, A)$ *is* $\mathcal{L}$*-rewritable*
- *there exists a* phantom $\Phi^k_{(\mathcal{O},A)}$ *in* $\mathcal{L}$*, for every* $k \in \mathbb{Z}$

Then, $(\mathcal{O}, \psi(\boldsymbol{t}))$ *is* $\mathcal{L}$*-rewritable.*

Proof. (Sketch) To simplify presentation, we assume that $\mathcal{O}$ does not contain $\bot$. The proof relies on the fact that, for any $\mathcal{A}$, there exists a canonical model $\mathcal{C}_{\mathcal{O},\mathcal{A}}$ of $(\mathcal{O}, \mathcal{A})$. Then, using the monotonicy of ψ, one can establish the following property, for all $\boldsymbol{\ell} \in \mathsf{tem}(\mathcal{A})$:

$$\boldsymbol{\ell} \in \mathsf{ans}(\boldsymbol{q}, \mathcal{A}) \qquad \text{iff} \qquad \mathcal{C}_{\mathcal{O},\mathcal{A}} \models \psi(\boldsymbol{\ell}). \tag{15}$$

The second important component of the proof is the fact that every OMPIQ $(\mathcal{O}, \varkappa)$ is $\mathcal{L}$-rewritable and has phantoms $\Phi^k_{(\mathcal{O},\varkappa)}$ in $\mathcal{L}$ if every OMAQ $(\mathcal{O}, A)$ with A from $\varkappa$ is such. This fact is shown by induction on the construction of $\varkappa$ using (15). As an example, we show how to construct a rewriting $\varphi_\varkappa(t)$ for $\varkappa = \Diamond_{\!F} A$ from a rewriting $\varphi_A(t)$ for A and phantoms $\Phi^k_{(\mathcal{O},A)}$. By the semantics of $\Diamond_{\!F}$, we could take

$$\varphi_\varkappa(t) = \exists t' \left((t' > t) \land \varphi_A(t')\right) \lor \bigvee_{k>0} \Phi^k_{(\mathcal{O},A)}.$$

Indeed, provided that 'infinite' formulas are allowed in rewritings, it is not hard to check, using (15), that $\varphi_\varkappa(t)$ is a rewriting of $(\mathcal{O}, \varkappa)$. To avoid the 'infinite' disjunction, we need an additional *periodicity* property of the $\mathit{LTL}^{\Box\bigcirc}_{horn}$ canonical models: for every ontology $\mathcal{O}$, there are positive integers $s_\mathcal{O}$ and $p_\mathcal{O}$ such that, for any data instance $\mathcal{A}$,

$$\boldsymbol{t}_\mathcal{O}(n) = \boldsymbol{t}_{\mathcal{O},\mathcal{A}}(n + p_\mathcal{O}), \text{ for } n \geq \max \mathcal{A} + s_\mathcal{O}.$$

In view of this periodicity, we can finitise the 'infinite' rewriting of $(\mathcal{O}, \varkappa)$ by taking

$$\varphi_\varkappa(t) = \exists t' \left((t' > t) \land \varphi_A(t')\right) \lor \bigvee_{0<k<s_\mathcal{O}+p_\mathcal{O}} \Phi^k_{(\mathcal{O},A)}.$$

The reader is invited to define phantoms $\Phi^k_{(\mathcal{O},\varkappa)}$ for $(\mathcal{O}, \varkappa)$ and $k > 0$ using the method above.

The third and last important component of this proof is that every monotone formula $\psi(\boldsymbol{t})$ is equivalent to a disjunction $\varphi(\boldsymbol{t}) = \bigvee_{l=1}^k \varphi_l(\boldsymbol{t})$ of formulas of the following form, for $l = 1, \dots, k$:

$$\varphi_l(\boldsymbol{t}) \;=\; \exists x_1, \dots, x_n \Big[\bigwedge_{i=1}^m (t_i = x_{j_i}) \land (x_1 < \dots < x_n) \land \bigwedge_{i=1}^n \alpha_i(x_i) \land \bigwedge_{i=1}^{n-1} \forall y \left((x_i < y < x_{i+1}) \to \beta_i(y)\right) \Big], \tag{16}$$

where the first conjunction contains $(t_i = x_1)$ and $(t_j = x_n)$ with free variables $t_i, t_j \in \boldsymbol{t}$, and the α_i and β_i are FO(<)-formulas with one free variable equivalent to some OMPIQs $(\mathcal{O}, \varkappa_i)$ and $(\mathcal{O}, \lambda_i)$ (respectively). This equivalence result is shown using Rabinovich's proof of Kamp's Theorem [74]. Let us assume

that $\psi(\boldsymbol{t})$ is a disjunction of formulas (16), and substitute in it each $\alpha_i(x_i)$ by $\varphi_{\varkappa_i}(x_i)$, where $\varphi_{\varkappa_i}(x_i)$ is a rewriting of $(\mathcal{O}, \varkappa_i)$, and similarly for $\beta_i(y)$. It is then straightforward to verify, relying on (15), that the result of this substitution is a rewriting of $(\mathcal{O}, \psi(\boldsymbol{t}))$. ❑

Corollary 21. *All monotone* $\mathrm{LTL}^{\Box\bigcirc}_{core}$ *OMQs are* $\mathrm{FO}(<, \equiv_{\mathbb{N}})$*-rewritable, and all monotone* $\mathrm{LTL}^{\Box}_{horn}$ *OMQs are* $\mathrm{FO}(<)$*-rewritable.*

4 OBDA with Temporalised *DL-Lite*

The OMQs considered so far are one-dimensional. For example, the ontology from Exercise 8 captures background knowledge about temporal events that can happen with one article submitted to a journal. If we want to use OBDA in order to query data about other articles, authors, co-authors, editors, etc., we need a language that is capable of representing knowledge about a second dimension, a domain populated by those articles, authors, co-authors, editors, and relationships between them.

Description logics (DLs) [12,13] form a prominent family of knowledge representation formalisms designed specifically for atemporal domains of that sort. However, far from all of the existing DLs are suitable for OBDA because OMQs with DL ontologies are not necessarily rewritable to standard target query languages such as FO (which is essentially SQL) or datalog; for more details we refer the reader to the course [61]. The *DL-Lite* family of DLs was developed for OBDA with relational databases and with the aim of guaranteeing FO-rewritability of all OMQs with conjunctive queries [73,29,6,84]. *DL-Lite* logics also underpin the *OWL 2 QL* profile of the Web Ontology Language OWL 2 standardised by the W3C.

We illustrate *DL-Lite* by a simple example (borrowed from [21]); for a detailed introduction to OBDA with *OWL 2 QL*, we refer the reader to [59,61].

Example 22. Consider the following ontology $\mathcal{O}$ given in the DL syntax and, for the reader's convenience, translated into first-order logic:

$$\begin{aligned}
&\mathit{ProjectManager} \sqsubseteq \exists \mathit{isAssistedBy}.\mathit{PA} \\
&\qquad \forall x \, \big(\mathit{ProjectManager}(x) \to \exists y \, (\mathit{isAssistedBy}(x, y) \land \mathit{PA}(y))\big) \\
&\exists \mathit{managesProject} \sqsubseteq \mathit{ProjectManager} \\
&\qquad \forall x \, \big(\exists y \, \mathit{managesProject}(x, y) \to \mathit{ProjectManager}(x)\big) \\
&\mathit{ProjectManager} \sqsubseteq \mathit{Staff} \\
&\qquad \forall x \, \big(\mathit{ProjectManager}(x) \to \mathit{Staff}(x)\big) \\
&\mathit{PA} \sqsubseteq \mathit{Secretary} \\
&\qquad \forall x \, \big(\mathit{PA}(x) \to \mathit{Secretary}(x)\big)
\end{aligned}$$

Here, *ProjectManager*, *PA*, *Staff*, and *Secretary* are *concept names* (corresponding to the unary predicates in the FO-translations), while *isAssistedBy* as well

as *managesProject* are *role names* (corresponding to the binary predicates). The conjunctive query

$$\psi(x) \;=\; \exists y \, \big(\mathit{Staff}(x) \land \mathit{isAssistedBy}(x, y) \land \mathit{Secretary}(y) \big)$$

asks for the members of staff that are assisted by secretaries. We invite the reader to verify that the following FO-query

$$\begin{aligned} \boldsymbol{Q}(x) \;=\; \exists y \, \big[\mathit{Staff}(x) \land \mathit{isAssistedBy}(x, y) \land (\mathit{Secretary}(y) \lor \mathit{PA}(y)) \big] \lor \\ \mathit{ProjectManager}(x) \lor \exists z \, \mathit{managesProject}(x, z) \end{aligned}$$

is an *FO-rewriting* of the OMQ $\boldsymbol{Q}(x) = (\mathcal{O}, \psi(x))$ in the sense that, for any data instance $\mathcal{A}$ (which contains ground atoms such as $\mathit{ProjectManager}(\mathit{bob})$, $\mathit{PA}(\mathit{joe})$ and $\mathit{isAssistedBy}(\mathit{bob}, \mathit{sam})$) and any individual name a from $\mathcal{A}$, we have $\mathcal{O}, \mathcal{A} \models \psi(a)$ iff $\boldsymbol{Q}(a)$ is true when evaluated directly over $\mathcal{A}$.

Being able to speak about domain concepts and roles, DL cannot say anything about their evolution in time. Following [8], we next introduce two-dimensional combinations of the *LTL* ontology languages from the previous section and various species of *DL-Lite*. The 2D language now contains *individual names* $a_0, a_1, \dots$, *concept names* $A_0, A_1, \dots$, and *role names* $P_0, P_1, \dots$. *Roles* S, *temporalised roles* R, *basic concepts* B, and *temporalised concepts* C are defined by the following grammars:

$$\begin{array}{lllllllllllllll} S & ::= & P_i & | & P_i^-, & R & ::= & S & | & \Box_{\!F} R & | & \Box_{\!P} R & | & \bigcirc_{\!F} R & | & \bigcirc_{\!P} R, \\ B & ::= & A_i & | & \exists S, & C & ::= & B & | & \Box_{\!F} C & | & \Box_{\!P} C & | & \bigcirc_{\!F} C & | & \bigcirc_{\!P} C. \end{array}$$

A *concept* or *role inclusion* takes the form

$$\vartheta_1 \sqcap \dots \sqcap \vartheta_k \;\sqsubseteq\; \vartheta_{k+1} \sqcup \dots \sqcup \vartheta_{k+m}, \tag{17}$$

where the ϑ_i are all temporalised concepts of the form C or, respectively, temporalised roles of the form R. When it does not matter whether we talk about concepts or roles, we refer to the ϑ_i as *terms*. A *TBox* $\mathcal{T}$ and an *RBox* $\mathcal{R}$ are finite sets of concept inclusions (CIs, for short) and, respectively, role inclusions (RIs, for short); their union $\mathcal{O} = \mathcal{T} \cup \mathcal{R}$ is called an *ontology*.

As before, we classify ontologies by the form of their inclusions and the temporal operators that occur in them. Let $\boldsymbol{c}, \boldsymbol{r} \in \{\mathit{bool}, \mathit{horn}, \mathit{krom}, \mathit{core}\}$ and $\boldsymbol{o} \in \{\Box, \bigcirc, \Box\bigcirc\}$. We denote by $\mathit{DL\text{-}Lite}^{\boldsymbol{o}}_{\boldsymbol{c}/\boldsymbol{r}}$ the *temporal description logic* whose TBoxes and RBoxes contain concept and role inclusions (17) satisfying $\boldsymbol{c}$ and $\boldsymbol{r}$, respectively, and the only temporal operators that occur in them are indicated in $\boldsymbol{o}$. Whenever $\boldsymbol{c} = \boldsymbol{r}$, we use a single subscript, that is, $\mathit{DL\text{-}Lite}^{\boldsymbol{o}}_{\boldsymbol{c}} = \mathit{DL\text{-}Lite}^{\boldsymbol{o}}_{\boldsymbol{c}/\boldsymbol{c}}$.

Note that, unlike the standard atemporal *DL-Lite* logics [29,6], which can have various types of CIs but allow only *core* RIs (of the form $S_1 \sqsubseteq S_2$ and $S_1 \sqcap S_2 \sqsubseteq \bot$), here we treat CIs and RIs in a uniform way and impose restrictions on the clausal structure of CIs and RIs separately.

An *ABox* (*data instance*, in DL parlance), $\mathcal{A}$, is a finite set of atoms of the form $A_i(a, \ell)$ and $P_i(a, b, \ell)$, where a and b are individual names and $\ell \in \mathbb{Z}$.

We denote by $\mathsf{ind}(\mathcal{A})$ the set of individual names in $\mathcal{A}$; as before, we also set $\mathsf{tem}(\mathcal{A}) = \{\, n \in \mathbb{Z} \mid \min \mathcal{A} \leq n \leq \max \mathcal{A} \,\}$. A *DL-Lite*$^{\boldsymbol{o}}_{\boldsymbol{c/r}}$ *knowledge base* (KB) is a pair $(\mathcal{O}, \mathcal{A})$, where $\mathcal{O}$ is a *DL-Lite*$^{\boldsymbol{o}}_{\boldsymbol{c/r}}$ ontology and $\mathcal{A}$ an ABox. The *size* $|\mathcal{O}|$ of an ontology $\mathcal{O}$ is the number of occurrences of symbols in $\mathcal{O}$; the size of a TBox, RBox, ABox, and knowledge base is defined analogously.

A (*temporal*) *interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}(n)})$, where $\Delta^{\mathcal{I}} \neq \emptyset$ and, for each $n \in \mathbb{Z}$,

$$\mathcal{I}(n) = (\Delta^{\mathcal{I}}, a_0^{\mathcal{I}}, \dots, A_0^{\mathcal{I}(n)}, \dots, P_0^{\mathcal{I}(n)}, \dots) \tag{18}$$

is a standard (atemporal) DL interpretation with $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, $A_i^{\mathcal{I}(n)} \subseteq \Delta^{\mathcal{I}}$ and $P_i^{\mathcal{I}(n)} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Thus, we assume that the domain $\Delta^{\mathcal{I}}$ and the interpretations $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ of the individual names are the same for all $n \in \mathbb{Z}$. (However, we do not adopt the unique name assumption.) The description logic and temporal constructs are interpreted in $\mathcal{I}(n)$ as follows:

$$\begin{array}{rcl@{\qquad}rcl}
(P_i^-)^{\mathcal{I}(n)} & = & \{\, (u,v) \mid (v,u) \in P_i^{\mathcal{I}(n)} \,\}, & & & \\
(\exists S)^{\mathcal{I}(n)} & = & \{\, u \mid (u,v) \in S^{\mathcal{I}(n)}, \text{ for some } v \,\}, & & & \\
(\Box_{\scriptscriptstyle F} \vartheta)^{\mathcal{I}(n)} & = & \bigcap\limits_{k>n} \vartheta^{\mathcal{I}(k)}, & (\Box_{\scriptscriptstyle P} \vartheta)^{\mathcal{I}(n)} & = & \bigcap\limits_{k<n} \vartheta^{\mathcal{I}(k)}, \\
(\bigcirc_{\scriptscriptstyle F} \vartheta)^{\mathcal{I}(n)} & = & \vartheta^{\mathcal{I}(n+1)}, & (\bigcirc_{\scriptscriptstyle P} \vartheta)^{\mathcal{I}(n)} & = & \vartheta^{\mathcal{I}(n-1)}.
\end{array}$$

As usual, $\bot$ is interpreted by $\emptyset$, and $\top$ by $\Delta^{\mathcal{I}}$ for concepts and by $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for roles. CIs and RIs are interpreted in $\mathcal{I}$ *globally*, i.e., (17) holds in $\mathcal{I}$ if

$$\vartheta_1^{\mathcal{I}(n)} \cap \dots \cap \vartheta_k^{\mathcal{I}(n)} \;\subseteq\; \vartheta_{k+1}^{\mathcal{I}(n)} \cup \dots \cup \vartheta_{k+m}^{\mathcal{I}(n)}, \qquad \text{for } \textit{all } n \in \mathbb{Z}.$$

Given an inclusion α, we write $\mathcal{I} \models \alpha$ if α holds in $\mathcal{I}$. We call $\mathcal{I}$ a *model* of $(\mathcal{O}, \mathcal{A})$ and write $\mathcal{I} \models (\mathcal{O}, \mathcal{A})$ if

$$\mathcal{I} \models \alpha, \text{ for all } \alpha \in \mathcal{O}, \qquad \begin{array}{l} a^{\mathcal{I}} \in A^{\mathcal{I}(\ell)}, \text{ for all } A(a,\ell) \in \mathcal{A}, \\ \qquad \text{and } \; (a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}(\ell)}, \text{ for all } P(a,b,\ell) \in \mathcal{A}. \end{array}$$

We say that $\mathcal{O}$ is *consistent* if there is an interpretation $\mathcal{I}$, a *model of* $\mathcal{O}$, such that $\mathcal{I} \models \alpha$, for all $\alpha \in \mathcal{O}$; we also say and that $\mathcal{A}$ is *consistent with* $\mathcal{O}$ if there is a model of $(\mathcal{O}, \mathcal{A})$. For an inclusion α, we write $\mathcal{O} \models \alpha$ if $\mathcal{I} \models \alpha$ for every model $\mathcal{I}$ of $\mathcal{O}$. A concept C is *consistent with* $\mathcal{O}$ if there is a model $\mathcal{I}$ of $\mathcal{O}$ and $n \in \mathbb{Z}$ such that $C^{\mathcal{I}(n)} \neq \emptyset$; consistency of roles with $\mathcal{O}$ is defined analogously.

We now define a language for querying temporal knowledge bases, which was inspired by the SPARQL 1.1 entailment regimes [44]; see also [69,46]. The main ingredients of the language are *positive temporal concepts* $\varkappa$ and *positive temporal roles* ϱ given by the following grammars:

$$\begin{array}{rcl}
\varkappa & ::= & \top \;\mid\; A_k \;\mid\; \exists S.\varkappa \;\mid\; \varkappa_1 \sqcap \varkappa_2 \;\mid\; \varkappa_1 \sqcup \varkappa_2 \;\mid\; \boldsymbol{op}_1\, \varkappa \;\mid\; \varkappa_1\, \boldsymbol{op}_2\, \varkappa_2, \\
\varrho & ::= & S \;\mid\; \varrho_1 \sqcap \varrho_2 \;\mid\; \varrho_1 \sqcup \varrho_2 \;\mid\; \boldsymbol{op}_1\, \varrho \;\mid\; \varrho_1\, \boldsymbol{op}_2\, \varrho_2,
\end{array}$$

where $\boldsymbol{op}_1 \in \{\bigcirc_F, \Diamond_F, \Box_F, \bigcirc_P, \Diamond_P, \Box_P\}$ and $\boldsymbol{op}_2 \in \{\mathcal{U}, \mathcal{S}\}$. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}(n)})$ be an interpretation. The *extensions* $\varkappa^{\mathcal{I}(n)}$ of $\varkappa$ in $\mathcal{I}$, for $n \in \mathbb{Z}$, are computed using the definition above and the following items:

$$
\begin{aligned}
&(\exists S.\varkappa)^{\mathcal{I}(n)} = \big\{\, u \in \Delta^{\mathcal{I}} \mid (u, v) \in S^{\mathcal{I}(n)}, \text{ for some } v \in \varkappa^{\mathcal{I}(n)} \,\big\},\\
&(\varkappa_1 \sqcap \varkappa_2)^{\mathcal{I}(n)} = \varkappa_1^{\mathcal{I}(n)} \cap \varkappa_2^{\mathcal{I}(n)}, \qquad (\varkappa_1 \sqcup \varkappa_2)^{\mathcal{I}(n)} = \varkappa_1^{\mathcal{I}(n)} \cup \varkappa_2^{\mathcal{I}(n)},\\
&(\Diamond_F \varkappa)^{\mathcal{I}(n)} = \bigcup_{k>n} \varkappa^{\mathcal{I}(k)}, \qquad (\Diamond_P \varkappa)^{\mathcal{I}(n)} = \bigcup_{k<n} \varkappa^{\mathcal{I}(k)},\\
&(\varkappa_1 \mathcal{U} \varkappa_2)^{\mathcal{I}(n)} = \bigcup_{k>n} \big(\varkappa_2^{\mathcal{I}(k)} \cap \bigcap_{n<m<k} \varkappa_1^{\mathcal{I}(m)}\big),\\
&(\varkappa_1 \,\mathcal{S}\, \varkappa_2)^{\mathcal{I}(n)} = \bigcup_{k<n} \big(\varkappa_2^{\mathcal{I}(k)} \cap \bigcap_{k<m<n} \varkappa_1^{\mathcal{I}(m)}\big).
\end{aligned}
$$

The definition of $\varrho^{\mathcal{I}(n)}$ is analogous. Note that positive temporal concepts $\varkappa$ and roles ϱ include all temporalised concepts C and roles R, respectively ($\exists S$ is a shortcut for $\exists S.\top$).

A *DL-Lite*$^{\boldsymbol{o}}_{\boldsymbol{c/r}}$ *ontology-mediated instance query* (OMIQ) is a pair of the form $\boldsymbol{q} = (\mathcal{O}, \varkappa)$ or $\boldsymbol{q} = (\mathcal{O}, \varrho)$, where $\mathcal{O}$ is a *DL-Lite*$^{\boldsymbol{o}}_{\boldsymbol{c/r}}$ ontology, $\varkappa$ is a positive temporal concept and ϱ a positive temporal role (which can use all temporal operators, not necessarily only those in $\boldsymbol{o}$). If $\varkappa$ is a basic concept (i.e., A or $\exists S$) and ϱ a role, then we refer to $\boldsymbol{q}$ as an *ontology-mediated atomic query* (OMAQ, as before). A *certain answer* to an OMIQ $(\mathcal{O}, \varkappa)$ over an ABox $\mathcal{A}$ is a pair $(a, \ell) \in \mathsf{ind}(\mathcal{A}) \times \mathsf{tem}(\mathcal{A})$ such that $a^{\mathcal{I}} \in \varkappa^{\mathcal{I}(\ell)}$ for every model $\mathcal{I}$ of $(\mathcal{O}, \mathcal{A})$. A *certain answer* to $(\mathcal{O}, \varrho)$ over $\mathcal{A}$ is a triple $(a, b, \ell) \in \mathsf{ind}(\mathcal{A}) \times \mathsf{ind}(\mathcal{A}) \times \mathsf{tem}(\mathcal{A})$ such that $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in \varrho^{\mathcal{I}(\ell)}$ for every model $\mathcal{I}$ of $(\mathcal{O}, \mathcal{A})$. The set of all certain answers to $\boldsymbol{q}$ over $\mathcal{A}$ is denoted, as before, by $\mathsf{ans}(\boldsymbol{q}, \mathcal{A})$.

Let $\mathcal{A}$ be any given ABox with $\mathsf{ind}(\mathcal{A}) = \{a_1, \dots, a_m\}$. We always assume, without much loss of generality, that $|\mathsf{tem}(\mathcal{A})| \geq |\mathsf{ind}(\mathcal{A})|$ (if this is not the case, we can simply add the required number of dummies with the missing time instances to $|\mathsf{tem}(\mathcal{A})|$). We represent $\mathcal{A}$ as a structure $\mathfrak{S}_{\mathcal{A}}$ with domain $\mathsf{tem}(\mathcal{A})$ ordered by $<$ such that

$$
\mathfrak{S}_{\mathcal{A}} \models A(k, \ell) \quad \text{iff} \quad A(a_k, \ell) \in \mathcal{A}, \qquad \mathfrak{S}_{\mathcal{A}} \models P(k, k', \ell) \quad \text{iff} \quad P(a_k, a_{k'}, \ell) \in \mathcal{A},
$$

for any concept and role names A, P and any $k, k', \ell \in \mathsf{tem}(\mathcal{A})$. To simplify notation, we often identify an individual name $a_k \in \mathsf{ind}(\mathcal{A})$ with its number representation $k \in \mathsf{tem}(\mathcal{A})$. The structure $\mathfrak{S}_{\mathcal{A}}$ represents a temporal database over which we can evaluate first-order formulas (queries) with predicates of the form $A(x, t)$, $P(x, y, t)$ and $t_1 < t_2$ as well as other standard numeric predicates and relational primitive recursion.

Definition 23 (FO-rewritability)**.** Let $\mathcal{L}$ be one of the three languages FO($<$), FO($<, \equiv_{\mathbb{N}}$) or FO(RPR). Let $\boldsymbol{q} = (\mathcal{O}, \varkappa)$ be an OMIQ and $\boldsymbol{Q}(x, t)$ a constant-free $\mathcal{L}$-formula with free variables x and t. We call $\boldsymbol{Q}(x, t)$ an *$\mathcal{L}$-rewriting of* $\boldsymbol{q}$ if, for any ABox $\mathcal{A}$, any $a \in \mathsf{ind}(\mathcal{A})$ and any $\ell \in \mathsf{tem}(\mathcal{A})$, we have $(a, \ell) \in \mathsf{ans}(\boldsymbol{q}, \mathcal{A})$ iff

$\mathfrak{S}_{\mathcal{A}} \models \boldsymbol{Q}(a, \ell)$. Similarly, a constant-free $\mathcal{L}$-formula $\boldsymbol{Q}(x, y, t)$ is an *$\mathcal{L}$-rewriting of* an OMIQ $\boldsymbol{q} = (\mathcal{O}, \varrho)$ if, for any ABox $\mathcal{A}$, any $a, b \in \mathsf{ind}(\mathcal{A})$ and any $\ell \in \mathsf{tem}(\mathcal{A})$, we have $(a, b, \ell) \in \mathsf{ans}(\boldsymbol{q}, \mathcal{A})$ iff $\mathfrak{S}_{\mathcal{A}} \models \boldsymbol{Q}(a, b, \ell)$.

Having all the definitions given, it is time to recall that two-dimensional combinations of knowledge representation formalisms can be computationally very complex if any non-trivial interaction between the dimensions is available; see, e.g., [41,54,65,42,43]. Our temporalised *DL-Lite* logics are not an exception as the following theorem demonstrates:

Theorem 24. (*i*) *Consistency checking for DL-Lite$^{\bigcirc}_{bool}$ KBs is undecidable.*

(*ii*) *There are DL-Lite$^{\bigcirc}_{bool}$ OMAQs $\boldsymbol{q}_1 = (\mathcal{O}, A)$ and $\boldsymbol{q}_2 = (\mathcal{O}, S)$ such that the problems whether, for a given ABox $\mathcal{A}$, the pair $(a, 0)$ is a certain answer to $\boldsymbol{q}_1$ over $\mathcal{A}$ and $(a, b, 0)$ is a certain answer to $\boldsymbol{q}_2$ over $\mathcal{A}$ are undecidable.*

Proof. (Sketch) (*i*) The proof is by reduction of the undecidable $\mathbb{N} \times \mathbb{N}$-tiling problem [19]. Given a set $\mathfrak{T} = \{1, \dots, k\}$ of tile types, we define a *DL-Lite$^{\bigcirc}_{bool}$* KB $(\mathcal{O}_{\mathfrak{T}}, \{I(a, 0)\})$ with the following ontology $\mathcal{O}_{\mathfrak{T}}$, where the R_i are role names associated with the tile types $i \in \mathfrak{T}$ and $\mathit{top}(i)$, $\mathit{bot}(i)$, $\mathit{right}(i)$, $\mathit{left}(i)$ are the colours on the four edges of any tile type i:

$$I \sqsubseteq \bigsqcup_{i \in \mathfrak{T}} \exists R_i, \quad R_i \sqsubseteq \bigsqcup_{\mathit{right}(i) = \mathit{left}(j)} \bigcirc_F R_j,$$
$$\exists R_i^- \sqsubseteq \bigsqcup_{\mathit{top}(i) = \mathit{bot}(j)} \exists R_j, \quad \exists R_i \sqcap \exists R_j \sqsubseteq \bot, \quad \text{for } i \neq j.$$

It is readily checked that $\{I(a, 0)\}$ is consistent with $\mathcal{O}_{\mathfrak{T}}$ iff $\mathfrak{T}$ can tile the $\mathbb{N} \times \mathbb{N}$ grid.

(*ii*) Using the representation of the universal Turing machine by means of tiles (see, e.g., [23]), we obtain a set $\mathfrak{U}$ of tile types for which the following problem is undecidable: given a finite sequence of tile types $i_0, \dots, i_n$, decide whether $\mathfrak{U}$ can tile the $\mathbb{N} \times \mathbb{N}$ grid so that tiles of types $i_0, \dots, i_n$ are placed on $(0, 0), \dots, (n, 0)$, respectively. Given such $i_0, \dots, i_n$, we take the ABox $\mathcal{A} = \{ I(a, 0), R_{i_0}(a, b, 0), \dots, R_{i_n}(a, b, n) \}$. Then $\mathfrak{U}$ can tile $\mathbb{N} \times \mathbb{N}$ with $i_0, \dots, i_n$ on the first row iff $\mathcal{A}$ is consistent with $\mathcal{O}_{\mathfrak{U}}$ iff $A(a, 0)$ is *not* a certain answer to OMAQ $(\mathcal{O}_{\mathfrak{U}}, A)$ over $\mathcal{A}$, where A is a fresh concept name; similar considerations apply to the case of a fresh role S. ❑

We can obtain better-behaved logics by restricting the expressive power of role inclusions. The proof of the following result can be found in [60].

Theorem 25. *Consistency checking for DL-Lite$^{\bigcirc}_{bool/krom}$, DL-Lite$^{\Box\bigcirc}_{bool/horn}$ and DL-Lite$^{\Box\bigcirc}_{horn}$ is* ExpSpace*-complete for combined complexity; for DL-Lite$^{\Box\bigcirc}_{bool/core}$ and DL-Lite$^{\Box\bigcirc}_{horn/core}$, it is* PSpace*-complete.*

Let us now consider FO-rewritability of OMQs with ontologies in temporalised *DL-Lite*. To avoid many a technical detail and concentrate on the main ideas, we only discuss here OMQs with atomic queries (that is, OMAQs) and

ontologies without occurrences of $\bot$ (which, therefore, are always consistent). Let $\mathcal{L}$ be one of the target languages FO($<$), FO($<, \equiv_{\mathbb{N}}$) or FO(RPR). We show that $\mathcal{L}$-rewritability of a $\bot$-free OMAQ with an *DL-Lite*$^{\Box\bigcirc}_{bool/horn}$ ontology is reducible to $\mathcal{L}$-rewritability of a role-free OMAQ. Ontologies without roles are clearly a notational variant of *LTL* ontologies; hence, in this case we can write '*LTL*$^{\Box\bigcirc}_{bool}$ ontologies'. We explain the reduction by instructive examples. The first two of them illustrate the interaction between the DL and temporal dimensions in *DL-Lite*$^{\Box\bigcirc}_{bool/horn}$ that we need to take into account when constructing the *LTL* OMAQs to which the rewritability of $\bot$-free *DL-Lite*$^{\Box\bigcirc}_{bool/horn}$ OMAQs is reduced.

Example 26. Suppose $\mathcal{T} = \{\, B \sqsubseteq \exists P,\ \exists Q \sqsubseteq A \,\}$ and $\mathcal{R} = \{\, P \sqsubseteq \bigcirc_{\!F} Q \,\}$. An obvious idea of constructing a rewriting for the OMAQ $\boldsymbol{q} = (\mathcal{T} \cup \mathcal{R},\, A)$ would be to find first a rewriting of the *LTL* OMAQ $(\mathcal{T}^\dagger, A^\dagger)$ obtained from $(\mathcal{T}, A)$ by replacing the basic concepts $\exists P$ and $\exists Q$ with surrogate concept names $(\exists P)^\dagger = E_P$ and $(\exists Q)^\dagger = E_Q$, respectively. This would give us the FO-query $A(x,t) \vee E_Q(x,t)$. By restoring the intended meaning of E_Q, we would then obtain $A(x,t) \vee \exists y\, Q(x,y,t)$. The second step would be to rewrite, using the RBox $\mathcal{R}$, the atom $Q(x,y,t)$ into $Q(x,y,t) \vee P(x,y,t-1)$. However, the resulting formula

$$A(x,t) \vee \exists y \, \big(Q(x,y,t) \vee P(x,y,t-1)\big)$$

is not a rewriting of $\boldsymbol{q}$ as it does not return the certain answer $(a,1)$ over the ABox $\mathcal{A} = \{B(a,0), C(a,1)\}$ because so far we have not taken into account the CI $\exists P \sqsubseteq \bigcirc_{\!F} \exists Q$, which is a consequence of $\mathcal{R}$. If we now add the 'connecting axiom' $(\exists P)^\dagger \sqsubseteq \bigcirc_{\!F} (\exists Q)^\dagger$ to $\mathcal{T}^\dagger$, then in the first step we obtain $A(x,t) \vee E_Q(x,t) \vee E_P(x,t-1) \vee B(x,t-1)$, which gives us the correct FO($<$)-rewriting of $\boldsymbol{q}$:

$$A(x,t) \vee \exists y \, \big(Q(x,y,t) \vee P(x,y,t-1)\big) \vee \exists y\, P(x,y,t-1) \vee B(x,t-1).$$

Example 27. Let $\boldsymbol{q} = (\mathcal{T} \cup \mathcal{R},\, A)$ with $\mathcal{T} = \{\, \exists Q \sqsubseteq \Box_{\!P} A \,\}$ and

$$\mathcal{R} = \{P \sqsubseteq \Box_{\!F} P_1,\ T \sqsubseteq \Box_{\!F} T_1,\ T_1 \sqsubseteq \Box_{\!F} T_2,\ P_1 \sqcap T_2 \sqsubseteq Q\}.$$

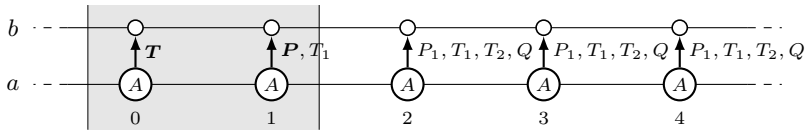
The two-step construction outlined in Example 26 would give us first the formula

$$\Phi(x,t) \ = \ A(x,t) \vee \exists t' \, \big((t < t') \wedge \exists y\, Q(x,y,t')\big)$$

as a rewriting of $(\mathcal{T}, A)$. It is readily checked that the following formula $\Psi(x,y,t')$ is a rewriting of $(\mathcal{R}, Q)$:

$$\begin{aligned} Q(x,y,t') \vee \big(\big[P_1(x,y,t') \vee \exists t'' \, \big((t'' < t') \wedge P(x,y,t'')\big)\big] \wedge \\ \big[T_2(x,y,t') \vee \exists t'' \, \big((t'' < t') \wedge \big(T_1(x,y,t'') \vee \\ \exists t''' \, \big((t''' < t'') \wedge T(x,y,t''')\big)\big)\big)\big]\big). \end{aligned}$$

However, the result of replacing $Q(x,y,t')$ in $\Phi(x,t)$ with $\Psi(x,y,t')$ is not an FO($<$)-rewriting of $(\mathcal{O}, A)$: when evaluated over $\mathcal{A} = \{\, T(a,b,0),\ P(a,b,1) \,\}$, it does not return the certain answers $(a,0)$ and $(a,1)$; see the picture below:



(Note that these answers would be found had we evaluated the obtained 'rewriting' over $\mathbb{Z}$ rather than $\{0,1\}$.) This time, we miss the concept inclusion $\exists(\Box_F P_1 \sqcap \Box_F T_2) \sqsubseteq \Box_F \exists Q$, which follows from $\mathcal{R}$ and $\mathcal{T}$. To fix the problem, we can take a fresh role name G_ρ, for $\rho = \{\, \Box_F P_1, \Box_F T_2 \,\}$, and add the 'connecting axiom' $\exists G_\rho \sqsubseteq \Box_F \exists Q$ to $\mathcal{T}$. Then, in the first step, we rewrite the extended TBox and A to the formula

$$\Phi'(x,t) \;=\; A(x,t) \vee \exists t' \big((t < t') \wedge \big(\exists y\, Q(x,y,t') \vee \exists t'' \big((t'' < t') \wedge \exists y\, G_\rho(x,y,t'')\big)\big)\big),$$

where we replace $Q(x,y,t')$ by $\Psi(x,y,t')$ and restore the meaning of $G_\rho(x,y,t')$ by rewriting $(\mathcal{R}, \Box_F P_1 \sqcap \Box_F T_2)$ to $P(x,y,t') \wedge \big(T_1(x,y,t') \vee \exists t'' \big((t'' < t) \wedge T(x,y,t'')\big)\big)$ and substituting it for $G_\rho(x,y,t')$ in $\Phi'(x,t)$.

We now formally define the connecting axioms for $\mathcal{O}$, assuming that $\mathcal{R}$ contains all role names in $\mathcal{T}$. Let ρ be a set of (temporalised) roles from $\mathcal{R}$ consistent with $\mathcal{R}$. Let $\boldsymbol{r}_\rho$ be the $\mathcal{R}$-canonical rod for $\{S(d,e,0) \mid S \in \rho\}$. Clearly, there are positive integers $s^\rho \leq |\mathcal{R}|$ and $p^\rho \leq 2^{2|\mathcal{R}|}$ with

$$\begin{aligned} &\boldsymbol{r}_\rho(n) = \boldsymbol{r}_\rho(n - p^\rho), && \text{for } n \leq -s^\rho, \\ &\boldsymbol{r}_\rho(n) = \boldsymbol{r}_\rho(n + p^\rho), && \text{for } n \geq s^\rho. \end{aligned}$$

Then we take a fresh role name G_ρ and fresh concept names D^n_ρ, for $-s^\rho - p^\rho < n < s^\rho + p^\rho$, and construct the CIs

$$\begin{gathered} \exists G_\rho \sqsubseteq D^0_\rho, \quad D^n_\rho \sqsubseteq \bigcirc_F D^{n+1}_\rho, \;\; \text{for } 0 \leq n < s^\rho + p^\rho - 1, \\ D^{s^\rho + p^\rho - 1}_\rho \sqsubseteq \bigcirc_F D^{s^\rho}_\rho, \\ D^n_\rho \sqsubseteq \exists S, \;\; \text{for } S \in \boldsymbol{r}_\rho(n),\; 0 \leq n < s^\rho + p^\rho, \end{gathered}$$

and symmetrical ones for $-s^\rho - p^\rho \leq n \leq 0$. Let **(con)** be the set of such CIs for all possible ρ. Set $\mathcal{T}_\mathcal{R} = \mathcal{T} \cup \textbf{(con)}$.

Example 28. In Example 26, for $\rho = \{P, \bigcirc_F Q\}$, we have $s^\rho = 2$, $p^\rho = 1$, and so $\mathcal{T}_\mathcal{R}$ contains the CIs

$$\begin{gathered} \exists P \sqsubseteq D^0_\rho, \quad D^0_\rho \sqsubseteq \bigcirc_F D^1_\rho, \quad D^1_\rho \sqsubseteq \bigcirc_F D^2_\rho, \\ D^2_\rho \sqsubseteq \bigcirc_F D^2_\rho, \quad D^0_\rho \sqsubseteq \exists P, \quad D^1_\rho \sqsubseteq \exists Q, \end{gathered}$$

which imply $\exists P \sqsubseteq \bigcirc_F \exists Q$. In the context of Example 27, for $\rho = \{\Box_F P_1, \Box_F T_2\}$, we have $s^\rho = 1$, $p^\rho = 1$, and so $\mathcal{T}_\mathcal{R}$ contains the following CIs:

$$\begin{gathered} \exists G_\rho \sqsubseteq D^0_\rho, \quad D^0_\rho \sqsubseteq \bigcirc_F D^1_\rho, \quad D^1_\rho \sqsubseteq \bigcirc_F D^1_\rho, \\ D^1_\rho \sqsubseteq \exists P_1, \quad D^1_\rho \sqsubseteq \exists T_2, \quad D^1_\rho \sqsubseteq \exists Q. \end{gathered}$$

We denote by $\mathcal{T}_{\mathcal{R}}^{\dagger}$ the $LTL_{bool}^{\Box\bigcirc}$ TBox obtained from $\mathcal{T}_{\mathcal{R}}$ by replacing every basic concept B with its surrogate $B^{\dagger}$.

Theorem 29. *Let $\mathcal{L}$ be one of* FO($<$), FO($<, \equiv_{\mathbb{N}}$), FO(RPR). *A DL-Lite$_{bool/horn}^{\Box\bigcirc}$ OMAQ $(\mathcal{O}, A)$ with a $\bot$-free $\mathcal{O} = \mathcal{T} \cup \mathcal{R}$ is $\mathcal{L}$-rewritable whenever*

- *the $LTL_{bool}^{\Box\bigcirc}$ OMAQ $(\mathcal{T}_{\mathcal{R}}^{\dagger}, A)$ is $\mathcal{L}$-rewritable and*
- *the $LTL_{horn}^{\Box\bigcirc}$ OMAQ $(\mathcal{R}, R)$ is $\mathcal{L}$-rewritable, for every temporalised role in $\mathcal{R}$.*

As a consequence of Theorems 13 and 29, we obtain:

Theorem 30. *Every DL-Lite$_{bool/horn}^{\Box\bigcirc}$ OMAQ is* FO(RPR)*-rewritable.*

Moreover, we also have the following:

Corollary 31. (*i*) *All DL-Lite$_{krom/core}^{\Box\bigcirc}$ OMAQs, and so all DL-Lite$_{core}^{\Box\bigcirc}$ OMAQs are* FO($<, \equiv_{\mathbb{N}}$)*-rewritable.*
(*ii*) *All DL-Lite$_{bool/horn}^{\Box\bigcirc}$ OMAQs are* FO(RPR)*-rewritable.*

These results can be extended to OMQs with more complex queries:

Theorem 32. (*i*) *All DL-Lite$_{horn/core}^{\Box}$ OMIQs are* FO($<$)*-rewritable.*
(*ii*) *All DL-Lite$_{core}^{\bigcirc}$ OMIQs are* FO($<, \equiv_{\mathbb{N}}$)*-rewritable.*
(*iii*) *All DL-Lite$_{horn}^{\Box\bigcirc}$ OMIQs are* FO(RPR)*-rewritable.*

It is to be noted that Theorem 32 holds for core and Horn *DL-Lite* logics only because even very simple disjunctive axioms lead to coNP-complete OMIQ answering problem. For example, as follows from [76], answering the atemporal *DL-Lite$_{krom}$* OMIQ

$$\big(\{\top \sqsubseteq A \sqcup B\},\ \exists R.(\exists P_1.A \sqcap \exists P_2.A \sqcap \exists N_1.B \sqcap \exists N_2.B)\big)$$

is coNP-complete for data complexity.

5 Ontology-Mediated Queries with *MTL* Ontologies

So far, we have discussed temporal ontologies with *LTL*-operators, which allowed us to naturally describe the behaviour of synchronous systems. Note, however, that *LTL*-representations of metric statements such as 'high temperature will be reached in at most 100 seconds' require long and unreadable disjunctions of the form

$$\bigcirc_F \mathit{HighTemp} \vee \bigcirc_F \bigcirc_F \mathit{HighTemp} \vee \bigcirc_F \bigcirc_F \bigcirc_F \mathit{HighTemp} \vee \dots \vee \underbrace{\bigcirc_F \dots \bigcirc_F}_{100} \mathit{HighTemp}.$$

Moreover, such formulas become meaningless in the case of asynchronous systems where time is dense and there is no next or previous moment.

A more suitable temporal knowledge representation formalism in this case is metric temporal logic *MTL*, which was introduced for modelling and reasoning about real-time systems [62,3]. Essentially, *MTL*-operators are obtained by

indexing the *LTL*-operators with arbitrary intervals ϱ, which allow concise representation of metric information and which can be used over dense time. We denote the metric counterparts of the future *LTL*-operators $\Diamond_F$, $\Box_F$, and $\mathcal{U}$ by $\Diamond_\varrho$, $\boxplus_\varrho$, and $\mathcal{U}_\varrho$, respectively; their past analogues are $\Diamond_\varrho$, $\boxminus_\varrho$, and $\mathcal{S}_\varrho$. Recall from Section 2 that, in the case of dense time, we assume that timestamps are dyadic rational numbers (whose set is denoted by $\mathbb{Q}_2$). Each interval ϱ, indexing an *MTL*-operator and called its *range*, has non-negative end-points from $\mathbb{Q}_2^{\geq 0}$ or ∞.

Example 33. The formula $\Diamond_{(0,100]} \mathit{HighTemp}$ is regarded to be true at a time instant $t \in \mathbb{Q}_2$ if *HighTemp* is true at some moment in the interval $(t, t + 100]$; dually, the formula $\boxplus_{(0,100]} \mathit{HighTemp}$ holds at t if *HighTemp* is true at every point of that interval. Note that the *LTL*-formula $\Diamond_F \mathit{HighTemp}$ has the same meaning as the *MTL*-formula $\Diamond_{(0,\infty)} \mathit{HighTemp}$ and $\Box_F \mathit{HighTemp}$ as $\boxplus_{(0,\infty)} \mathit{HighTemp}$.

Similarly to the case of *LTL*, we treat ontology axioms with *MTL*-operators as (variable-free) abbreviations for monadic first-order formulas. However, since now time is dense, instead of MFO(<)-formulas, we use MFO$(<, \delta_{\leq \mathbb{Q}_2}, \delta_{=\mathbb{Q}_2})$-formulas, where $\delta_{<a}$ and $\delta_{=a}$, for $a \in \mathbb{Q}_2^+$, are additional built-in binary predicates allowing us to speak about the distance between moments of time. Recall that $\delta_{<a}(x, y)$ stands for $0 \leq x - y < a$, and $\delta_{=a}(x, y)$ for $x - y = a$. Using these predicates, for every range ϱ, we can define a predicate $\delta_\varrho(x, y)$ saying that $x - y \in \varrho$; for example, we can set

$$\delta_{[1,2)}(x, y) = \delta_{\geq 1}(x, y) \wedge \delta_{<2}(x, y).$$

We can define the *standard translation* ‡ from *MTL* into MFO$(<, \delta_{\leq \mathbb{Q}_2}, \delta_{=\mathbb{Q}_2})$, similar to the translation † of *LTL* into MFO(<), by taking, for example

$$\begin{aligned} (\boxplus_\varrho \varkappa)^\ddagger &= \forall t' \left(\delta_\varrho(t', t) \to \varkappa^\ddagger\{t/t'\}\right), \\ (\Diamond_\varrho \varkappa)^\ddagger &= \exists t' \left(\delta_\varrho(t', t) \wedge \varkappa^\ddagger\{t/t'\}\right). \end{aligned}$$

As mentioned in Section 2, there are two semantics for MFO$(<, \delta_{\leq \mathbb{Q}_2}, \delta_{=\mathbb{Q}_2})$ and *MTL*. In the *continuous semantics*, the temporal domain in the definition (2) of $\mathsf{ans}(\boldsymbol{q}, \mathcal{A})$ is the whole of $\mathbb{Q}_2$; in the *pointwise semantics*, it coincides with the active domain $\mathsf{tem}(\mathcal{A})$. Under the former, MFO$(<, \delta_{\leq \mathbb{Q}_2}, \delta_{=\mathbb{Q}_2})$-formulas with one free variable have the same expressive power as *MTL*-formulas interpreted over the reals $\mathbb{R}$ [55]. On the other hand, the satisfiability and model checking problems turn out to be undecidable in this case [50]. However, as shown in [71], both problems become decidable, albeit with non-primitive recursive complexity, if the pointwise semantics is adopted.

Let us now turn to OBDA with *MTL*-ontologies. Compared to the *LTL* OMQs, research of the *MTL* case has just begun, and very few results are known at the moment. In particular, FO-rewritability of *MTL* OMQs under the continuous semantics is practically *terra incognita*; we refer the reader to [26,25,82,83] and the survey [9]. In the remainder of this section, we discuss what is known about FO-rewritability of *MTL* OMQs under the pointwise semantics following [75]. We begin with an example.

Example 34. Consider the *MTL*-ontology $\mathcal{O} = \{\boxminus_{[0,2)} B \to B',\ \diamondminus_{[1,1]} B' \to A\}$ and the data instances $\mathcal{A}_1 = \{B(0), B(1/2), C(3/2)\}$ and $\mathcal{A}_2 = \{B(0), C(3/2)\}$. Then $\mathsf{tem}(\mathcal{A}_1) = \{0, 1/2, 3/2\}$ and $\mathsf{tem}(\mathcal{A}_2) = \{0, 3/2\}$. The minimal models $\mathcal{I}_1$ of $(\mathcal{O}, \mathcal{A}_1)$ and $\mathcal{I}_2$ of $(\mathcal{O}, \mathcal{A}_2)$ are shown in the picture below:

Hence, $3/2$ is a certain answer to the OMAQ $(\mathcal{O}, A)$ over $\mathcal{A}_1$, but there is no certain answer to $(\mathcal{O}, A)$ over $\mathcal{A}_2$.

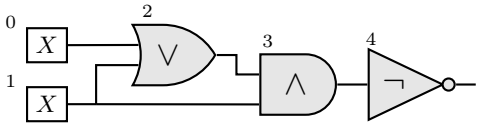
In what follows, we consider FO-rewritability of *MTL* OMAQs only. The following observation is simple and left to the reader.

Theorem 35. *Answering MTL OMAQs is in* coNP *for data complexity.*

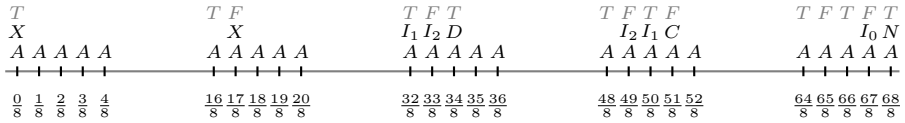
The next theorem establishes some lower complexity bounds for answering *MTL* OMQs, which should be compared with the data complexity of *LTL* OMQs. In this theorem, MTL^{-} means *MTL* with *past* operators only, with MTL^{-}_{horn} and MTL^{-}_{core} being its Horn and core fragments, respectively.

Theorem 36. (*i*) *Answering MTL OMAQs is* coNP*-hard for data complexity.*
(*ii*) *Answering OMAQs with an* MTL^{-}_{core} *ontology is* P*-hard for data complexity.*
(*iii*) *Answering OMAQs with an* MTL^{-}_{core} *ontology containing* $\diamondminus$ *operators only is* NL*-hard for data complexity.*

Proof. (Sketch) We show (*i*) by reduction of the complement of the NP-complete circuit satisfiability problem [5]. Instead of presenting the details of the construction, we only consider the Boolean circuit below, which contains two input gates, one AND gate, one OR gate, and one NOT gate.



First, we enumerate the gates of the circuit with consecutive numbers starting from 0 so that if there is an edge from n to m, then $n < m$ (observe that such an enumeration always exists). Then we define a data instance $\mathcal{A}$ stating that A holds in 25 timestamps. These timestamps form 5 sections, each containing 5 timestamps (as there are 5 gates in the circuit) as depicted below:



The data instance $\mathcal{A}$ also contains facts about X (for the input gates), D (for the OR gate), C (for the AND gate), N (for the NOT gate), I_0 (for the input of

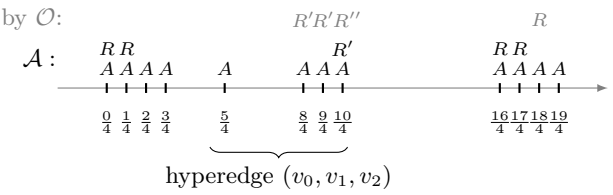
the NOT gate), and I_1 and I_2 (for the first and the second inputs to the binary gates, i.e., OR and AND). The i-th section of $\mathcal{A}$ represents information about the i-th gate. For example, the left-most section of $\mathcal{A}$ describes the gate 0 by stating that X holds in the left-most timestamp in this section. On the other hand, the third section from the left describes the OR gate by stating that D holds in the third from the left timestamp in this section (which expresses that the gate number 2 is an OR gate), that I_1 holds in the first timestamp from the left in the section (which expresses that the gate number 0 is the first input to the OR gate), and that I_2 holds in the second timestamp from the left in the section (which expresses that the gate number 1 is the second input to the OR gate).

We can now construct an ontology that allows checking unsatisfiability of any circuit. The ontology uses two more atoms T and F, which stand for 'true' and 'false' and which allow us to simulate propagation of signals in the circuit:

$$
\begin{array}{ll}
X \to T \lor F, & \Diamond_{[2,2]} T \to T, \qquad \Diamond_{[2,2]} F \to F, \\
N \land \Diamond_{[0,1]}(I_0 \land T) \to F, & N \land \Diamond_{[0,1]}(I_0 \land F) \to T, \\
D \land \Diamond_{[0,1]}(I_1 \land T) \to T, & D \land \Diamond_{[0,1]}(I_2 \land T) \to T, \\
C \land \Diamond_{[0,1]}(I_1 \land F) \to F, & C \land \Diamond_{[0,1]}(I_2 \land F) \to F, \\
D \land \Diamond_{[0,1]}(I_1 \land F) \land \Diamond_{[0,1]}(I_2 \land F) \to F, & \\
C \land \Diamond_{[0,1]}(I_1 \land T) \land \Diamond_{[0,1]}(I_2 \land T) \to T. &
\end{array}
$$

A possible propagation of T and F enforced by the ontology is shown in the previous picture with T and F written in grey. A given circuit is unsatisfiable iff the right-most timestamp in $\mathcal{A}$ is a certain answer to an OMAQ consisting of the above ontology and the query F.

(*ii*) The proof is by reduction of path system accessibility (PSA) which consists of checking whether there exists a hyperpath between two nodes in a hypergraph (the problem is well-known to be P-complete). For example, assume that a hypergraph contains vertices $0, 1, 2, 3$, and a single hyperedge $(0, 1, 2)$. Suppose we want to check whether the vertex $t = 3$ is accessible from the set of vertices $S = \{0, 1\}$, i.e., whether $t \in S$ or there are vertices u, w accessible from S and (u, w, t) is a hyperedge. For such an instance of PSA, we construct the following data instance $\mathcal{A}$:

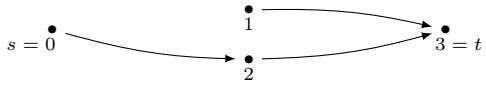


We next define an MTL^{-}_{core}-ontology $\mathcal{O}$ with the axioms:

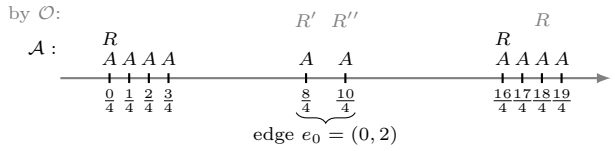
$$
\Diamond_{[2,2]} R \to R', \qquad \boxminus_{[0,1]} R' \to R'', \qquad \Diamond_{[2,2]} R'' \to R, \qquad \Diamond_{[4,4]} R \to R.
$$

Then $4k+t/N$ is a certain answer to $(\mathcal{O}, R)$ over $\mathcal{A}$ iff t is accessible from S in G.

We prove (*iii*) by reduction of the NL-complete reachability problem in acyclic digraphs. As an example, assume that we want to check whether $t = 3$ is accessible from $s = 0$ in the following directed graph G:



To this end, we construct a data instance $\mathcal{A}_G$, whose initial segment (corresponding to the edge from 0 to 2) is depicted below:



Let $\mathcal{O}$ be a MTL^{-}_{core}-ontology with the following rules:

$$\Diamond_{[2,2]} R \to R', \qquad \Diamond_{(0,1]} R' \to R'', \qquad \Diamond_{[2,2]} R'' \to R, \qquad \Diamond_{[4,4]} R \to R.$$

Then $4k+t/N$ is a certain answer to $(\mathcal{O}, R)$ over $\mathcal{A}$ iff t is reachable from s in G. ❑

In the next theorem, we show that, for each MTL^{-}_{horn}-OMAQ, there is a rewriting to datalog queries with additional FO-formulas built from EDB predicates in their rule bodies, answering which is in P [45].

Theorem 37. *Every MTL^{-}_{horn}-OMAQ is datalog*(FO)*-rewritable, and so can be answered in* P *for data complexity.*

Proof. (Sketch) To illustrate how one can construct a datalog(FO)-rewriting, we consider the MTL^{-}_{horn}-OMAQ $\boldsymbol{q} = (\mathcal{O}, \mathit{Alert})$, where $\mathcal{O}$ consists of the axiom:

$$\Diamond_{(0,600]} \mathit{RotorSpeedAbove1260} \land \boxminus_{[0,10]} \mathit{MainFlameOn} \to \mathit{Alert}.$$

We construct a datalog(FO)-rewriting (Π, G) of $\boldsymbol{q}$. We will use $\mathit{suc}(x, y)$ stating that x is the immediate successor of the timestamp y.

The first block of Π has the following rules for every atom B occurring in $\boldsymbol{q}$:

$$B(x) \to B'(x, x), \qquad B'(x, y) \land B'(z, z) \land \mathit{suc}(y, z) \to B'(x, z).$$

Intuitively, instead of unary predicates $B(x)$ stating that B holds at a timestamp x, we want to obtain binary predicates $B'(x, y)$ stating that B holds in the interval $[x, y]$ (i.e., in all timestamps from the temporal domain that belong to this interval). The first rule converts $B(x)$ into $B'(x, x)$, while the second rule allows us to coalesce intervals by stating that if B holds in $[x, y]$ and in z, and moreover, there is no timestamp between y and z, then B holds in $[x, z]$.

The second block of Π consists of rules obtained by modifying axioms in $\mathcal{O}$, where every atom B not in the scope of a metric operator is replaced by $B'(x,x)$, whereas atoms in the scope of metric operators are replaced by corresponding formulas using predicates such as $\delta_{<a}$ and $\delta_{>a}$. In particular, in our example the single rule in $\mathcal{O}$ would be replaced with:

$$\big(\mathit{MainFlameOn}'(w,z) \land \delta_{\geq 600}(x,w) \land \delta_{<0}(x,z) \land \delta_{\geq 600}(z,w)\big) \land \\ \big(\mathit{RotorSpeedAbove1260}'(w,z) \land \delta_{>0}(x,w) \land \delta_{\leq 6000}(x,z)\big) \to \mathit{Alert}'(x,x).$$

Finally, the program Π contains the rule

$$\mathit{Alert}'(x,x) \to G(x),$$

which states that if A holds at x, then G also holds at x.

Now, for every data instance $\mathcal{A}$, t is a certain answer to $\boldsymbol{q}$ over $\mathcal{A}$ iff t is an answer to (Π, G) over $\mathcal{A}$. As a result, (Π, G) is a datalog(FO)-rewriting of $\boldsymbol{q}$.

Note that so far we have assumed that $\mathcal{A}$ is in the signature (3). However, this assumption does not differentiate between small (e.g., 0.01) and large (e.g., 1000.00000001) timestamps in $\mathcal{A}$. Instead, we can consider $\mathcal{A}$ in a different signature using *binary* predicates $\mathit{bit}(i,j)$ indicating that the integer part of the timestamp i has bit j equal to 1, and analogous predicates for the fractional part. In this new representation of $\mathcal{A}$, we do not assume the predicates $\delta_{<a}(x,y)$ and $\delta_{=a}(x,y)$, however, we can express them as FO(<)-formulas $\varphi_{<a}(x,y)$ and $\varphi_{=a}(x,y)$ with the predicates *bit* (see [75] for details). The predicate $\mathit{suc}(x,y)$ is also expressible in the new representation as $(y < x) \land \neg\exists z\,(y < z < x)$. ❑

The next theorem establishes rewritability into FO(TC) that extends FO(<) with the transitive closure operator. In complexity-theoretic terms, FO(TC) corresponds to NL [56].

Theorem 38. *Every MTL^{-}_{core}-OMAQ with an ontology containing $\diamondminus_\varrho$ operators only is* FO(TC)*-rewritable, and so can be answered in* NL *for data complexity.*

Proof. (Sketch) Let $\boldsymbol{q} = (\mathcal{O}, A)$ be an MTL^{-}_{core}-OMAQ with $\diamondminus_\varrho$ operators only. First, we delete all disjointness constraints of the form $C_1 \land C_2 \to \bot$ from $\mathcal{O}$ and we translate the query into datalog(FO) in the same way as we did in the proof of Theorem 37. Since $\boldsymbol{q}$ is core, by the form of this translation, we obtain a linear datalog(FO) program. It is well-known that such a program can be transformed into an FO(TC)-query $\Psi_A(x)$ [45]. For every disjointness constraint $C_1 \land C_2 \to \bot$ in $\mathcal{O}$, we take the sentence $\exists x\,(\Psi_{C_1}(x) \land \Psi_{C_2}(x))$ and, finally, form a disjunction of $\Psi_A(x)$ with those sentences, which is an FO(TC)-rewriting of $\boldsymbol{q}$. ❑

The rewritability results we are going to present next impose restrictions on the range of metric operators: in particular, we consider ontologies with *unbounded* ranges of the form $\langle r, \infty)$, for $r \in \mathbb{Q}_2^{\geq 0}$ and $\langle$ being (or [, *punctual* ranges $[r,r]$, for $r \in \mathbb{Q}_2^{\geq 0}$, and *non-punctual* ranges.

Theorem 39. *Every MTL$^-$-OMAQ $\boldsymbol{q} = (\mathcal{O}, A)$ with temporal operators in $\mathcal{O}$ of the form*

- $\diamondminus_{\langle r,\infty)}$ *and* $\boxminus_{\langle r,\infty)}$ *only is* FO(<)*-rewritable, and so can be answered in* AC^0*;*
- $\diamondminus_{[r,r]}$ *and* $\boxminus_{[r,r]}$ *only is* FO(RPR)*-rewritable, and answering such OMAQs is* NC^1*-complete for data complexity;*
- $\diamondminus_{\varrho}$ *and* $\boxminus_{\varrho}$*, for non-punctual* ϱ*, only is* FO(TC)*-rewritable; answering such OMAQs is in* NL *and* NC^1*-hard for data complexity.*

Moreover, MTL^-_{horn}-OMAQs from the last item are also FO(DTC)-rewritable (FO with deterministic transitive closure), and so answering them is in L for data complexity. These rewritings heavily depend on the properties of the minimal models of OMAQs of a given type. For example, in the case of *MTL*-OMAQ with temporal operators of the forms $\diamondminus_{\langle r,\infty)}$ and $\boxminus_{\langle r,\infty)}$ only, we can observe that the minimal model is monotonic in the sense that if $\diamondminus_{\langle r,\infty)}P$ holds in a timestamp t, then the same formula holds in all timestamps greater than t. Analogously, if $\diamondminus_{\langle r,\infty)}P$ does not hold in a timestamp t, then this formula holds in all timestamps smaller than t. It turns out that we can use this observation (and some additional properties of the minimal model) to construct an FO(<)-rewriting. For more details of the constructions we refer the reader to [75].

So far in this section we discussed *MTL*-OMAQs. What do we know about wider classes of *MTL* OMQs (under the pointwise semantics)? First, we have:

Theorem 40. *Every MTL OMQ $\boldsymbol{q} = (\mathcal{O}, \psi(\boldsymbol{t}))$ with an* $\mathrm{MFO}(<, \delta_{\leq \mathbb{Q}_2}, \delta_{=\mathbb{Q}_2})$*-formula $\psi(\boldsymbol{t})$ is* coNP*-complete for data complexity.*

Note that the counterpart of this result in *LTL* is Theorem 13. Furthermore, we say that $\mathrm{MFO}(<, \delta_{\leq \mathbb{Q}_2}, \delta_{=\mathbb{Q}_2})$-formula is *positive*, if it is constructed using $\wedge$, $\vee$, $\forall$, $\exists$. We say that an *MTL*-OMQ $\boldsymbol{q} = (\mathcal{O}, \psi(\boldsymbol{t}))$ is positive, if $\psi(\boldsymbol{t})$ is a positive $\mathrm{MFO}(<, \delta_{\leq \mathbb{Q}_2}, \delta_{=\mathbb{Q}_2})$-formula. The following result follows from [75]:

Theorem 41. *Suppose $\boldsymbol{q} = (\mathcal{O}, \psi(\boldsymbol{t}))$ is a positive MTL_{horn}-OMQ and $\mathcal{L}$ a language containing* FO(<)*. Suppose also that, for each atomic A in ψ, the OMAQ $(\mathcal{O}, A)$ is $\mathcal{L}$-rewritable. Then, $\boldsymbol{q}$ is also $\mathcal{L}$-rewritable.*

Therefore, by Theorems 37, 38,and 39, we obtain:

Corollary 42. (*i*) *Every positive MTL^-_{horn}-OMQ is datalog*(FO)*-rewritable.*

(*ii*) *Every positive MTL^-_{core}-OMQ with an ontology containing $\diamondminus_{\varrho}$ operators only is* FO(TC)*-rewritable.*

(*iii*) *Every positive MTL$^-$-OMQ with an ontology containing $\diamondminus_{\langle r,\infty)}$ and $\boxminus_{\langle r,\infty)}$ operators only is* FO(<)*-rewritable.*

(*iv*) *Every positive MTL$^-$-OMQ with an ontology containing $\diamondminus_{[r,r]}$ and $\boxminus_{[r,r]}$ operators only is* FO(RPR)*-rewritable.*

(*v*) *Every positive MTL$^-$-OMQ with an ontology containing $\diamondminus_{\varrho}$ and $\boxminus_{\varrho}$ for non-punctual ϱ only is* FO(TC)*-rewritable.*

It is of interest to compare the results above to the results in the *LTL* case given by Corollary 21. The latter results also hold for positive OMQs, since every

positive OMQ is also monotone. The comparison shows that, even in the case of positive Horn OMQs, the complexity landscape is more diverse for *MTL* than for *LTL*.

6 Future Research

In this paper, we have tried to overview recent developments in ontology-based access to temporal data, focusing primarily on the problem of rewriting temporal ontology-mediated queries into first-order queries over the data. For other aspects of temporal OBDA the reader is referred to the survey [9]. Compared to the classical atemporal OBDA, its temporal counterpart is still an infant, from both theoretical and especially practical points of view. In the remainder of this concluding section, we briefly discuss a few open problems and directions for future research that are related to OBDA with *LTL*, *MTL* and temporal *DL-Lite*.

LTL. The classical OBDA theory has recently investigated the fine-grained combined and parameterised complexities of OMQ answering and the succinctness problem for FO-rewritings [21,20,22,15]. These problems are of great importance for the temporal case, too (in particular, because the rewritings discussed above are far from optimal). Another development in the classical OBDA theory is the classification of single ontologies and even OMQs according to their data complexity and rewritability [67,66,51]. Extending this approach to temporal OMQs will most probably require totally different methods because of the linearly ordered temporal domain.

Temporal DL-Lite. The technique considered above does not seem to work for (two-sorted) conjunctive queries in place of instance queries; on the other hand, the methods of [7] indicate a somewhat different approach to deal with not necessarily tree-shaped conjunctive queries having multiple answer variables. It is of interest what happens with OMAQs when we consider Krom role inclusions. There are two open questions: the upper bound for $\mathit{DL\text{-}Lite}^{\Box}_{bool}$ OMAQs and FO-rewritability of $\mathit{DL\text{-}Lite}^{\Box}_{krom}$ and $\mathit{DL\text{-}Lite}^{\Box\bigcirc}_{krom}$ OMAQs. We conjecture that $\mathit{DL\text{-}Lite}^{\Box}_{krom}$ OMAQs are FO($<$)-rewritable and $\mathit{DL\text{-}Lite}^{\Box\bigcirc}_{krom}$ OMAQs are FO($<, \equiv_{\mathbb{N}}$)-rewritable. To show this, one may need a type-based technique similar to the approach in the proof of Theorem 18 as Theorem 29 is not applicable to Krom role inclusions. Another open question is the upper bound for $\mathit{DL\text{-}Lite}^{\Box}_{bool}$ OMAQs. On the practical side, more real-world use cases are needed to understand which temporal constructs are required to specify relevant temporal events and evaluate the performance of OMQ rewritings; for some activities in this direction we refer the reader to [17,58,25,24].

MTL. Much less is known about OMQs with *MTL*-ontologies than about ontologies with *LTL*-operators. The results presented in this paper show that some *MTL*-ontologies can be rewritten to FO (and its extensions), which indicates that reasoning with such ontologies may be feasible in practise. However, the

computational properties of *MTL*-ontologies heavily depend on many aspects, such as the form of available metric operators and the type of ranges they use. The results presented in the previous section are only preliminary observations, and we are clearly far from understanding well the complexity of answering *MTL*-OMQs and classifying them accordingly. Here we mention some interesting open problems. Observe that Theorem 39 about OMAQs with non-punctual ranges contains a gap between NL and NC^1, and between L and NC^1. It would be interesting to establish tight complexity bounds for such OMAQs, as non-punctuality of ranges is a standard approach in *MTL*, which often allows one to reduce significantly the computational costs [2]. There are also many types of *MTL*-OMQs that have not been considered so far, for example, the ones with the 'since' and 'until' operators, as well as more complex queries, such as quasi-positive, or general (used in *LTL*-OMQs).

Another direction of future work concerns adopting the continuous semantics. Interestingly, such an approach has been already considered with respect to extensions of Datalog with *MTL*-operators [26,82] and applied, for example, to stream reasoning [83,49,48,63]. It is also worth mentioning that there is work on combining DLs with *MTL* [47,11]. As there is a big variety of potential applications of ontologies with *MTL*-operators, for instance, to image sequence evaluation [27], to ambient intelligence context [70,18], or to online monitoring asynchronous systems [53,37,14], among others, this direction of research looks very promising.

Acknowledgements

This work was supported by the EPSRC U.K. grants EP/S032282 '$quant^{MD}$: Ontology-Based Management for Many-Dimensional Quantitative Data' and EP/S032347 'OASIS: Ontology Reasoning over Frequently-Changing and Streaming Data', and by the SIRIUS Centre for Scalable Data Access (Research Council of Norway).

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. Journal of the ACM (JACM) 43(1), 116–146 (1996)
3. Alur, R., Henzinger, T.A.: Real-time logics: Complexity and expressiveness. Inf. Comput. 104(1), 35–77 (1993), http://dx.doi.org/10.1006/inco.1993.1025
4. Apt, K.R.: Logic programming. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, pp. 493–574. Elsevier and MIT Press (1990), https://doi.org/10.1016/b978-0-444-88074-1.50015-9
5. Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press, New York, NY, USA, 1st edn. (2009)
6. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. Journal of Artificial Intelligence Research (JAIR) 36, 1–69 (2009)

7. Artale, A., Kontchakov, R., Wolter, F., Zakharyaschev, M.: Temporal description logic for ontology-based data access. In: Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI). IJCAI/AAAI (2013)
8. Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: First-order rewritability of temporal ontology-mediated queries. In: Proc. of the 24th Int. Joint Conference on Artificial Intelligence, IJCAI'15. pp. 2706–2712 (2015)
9. Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: Ontology-mediated query answering over temporal data: A survey (invited talk). In: TIME. LIPIcs, vol. 90, pp. 1:1–1:37. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
10. Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: First-order rewritability of ontology-mediated queries in linear temporal logic. CoRR abs/2004.07221 (2020), https://arxiv.org/abs/2004.07221
11. Baader, F., Borgwardt, S., Koopmann, P., Ozaki, A., Thost, V.: Metric temporal description logics with interval-rigid names. In: International Symposium on Frontiers of Combining Systems. pp. 60–76. Springer (2017)
12. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press, Cambridge, UK, 2 edn. (2007)
13. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017)
14. Baldor, K., Niu, J.: Monitoring dense-time, continuous-semantics, metric temporal logic. In: International Conference on Runtime Verification. pp. 245–259. Springer (2012)
15. Barceló, P., Feier, C., Lutz, C., Pieris, A.: When is ontology-mediated querying efficient? CoRR abs/2003.07800 (2020), https://arxiv.org/abs/2003.07800
16. Barrington, D.A.M., Compton, K.J., Straubing, H., Thérien, D.: Regular languages in nc^1. J. Comput. Syst. Sci. 44(3), 478–499 (1992), https://doi.org/10.1016/0022-0000(92)90014-A
17. Beck, H., Dao-Tran, M., Eiter, T., Fink, M.: LARS: A logic-based framework for analyzing reasoning over streams. In: Bonet, B., Koenig, S. (eds.) Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. pp. 1431–1438. AAAI Press (2015), http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9657
18. Bellotto, N., Benfold, B., Harland, H., Nagel, H.H., Pirlo, N., Reid, I., Sommerlade, E., Zhao, C.: Cognitive visual tracking and camera control. Computer Vision and Image Understanding 116(3), 457–471 (2012)
19. Berger, R.: The Undecidability of the Domino Problem. American Mathematical Society (1966)
20. Bienvenu, M., Kikot, S., Kontchakov, R., Podolskii, V.V., Ryzhikov, V., Zakharyaschev, M.: The complexity of ontology-based data access with OWL 2 QL and bounded treewidth queries. In: Proc. of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017. pp. 201–216. ACM (2017)
21. Bienvenu, M., Kikot, S., Kontchakov, R., Podolskii, V.V., Zakharyaschev, M.: Ontology-mediated queries: Combined complexity and succinctness of rewritings via circuit complexity. J. ACM 65(5), 28:1–28:51 (2018)

22. Bienvenu, M., Kikot, S., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: On the parametrised complexity of tree-shaped ontology-mediated queries in OWL 2 QL. In: Artale, A., Glimm, B., Kontchakov, R. (eds.) Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017. CEUR Workshop Proceedings, vol. 1879. CEUR-WS.org (2017), http://ceur-ws.org/Vol-1879/paper55.pdf
23. Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. Perspectives in Mathematical Logic, Springer (1997)
24. Brandt, S., Calvanese, D., Kalaycı, E.G., Kontchakov, R., Mörzinger, B., Ryzhikov, V., Xiao, G., Zakharyaschev, M.: Two-dimensional rule language for querying sensor log data: a framework and use cases. In: Proc. of the 26th Int. Symposium on Temporal Representation and Reasoning, TIME 19. vol. 147, pp. 7:1–7:15. Dagstuhl Publishing (2019)
25. Brandt, S., Kalayci, E.G., Ryzhikov, V., Xiao, G., Zakharyaschev, M.: Querying log data with metric temporal logic. J. Artif. Intell. Res. 62, 829–877 (2018)
26. Brandt, S., Kalaycı, E.G., Ryzhikov, V., Xiao, G., Zakharyaschev, M.: Querying log data with metric temporal logic. Journal of Artificial Intelligence Research 62, 829–877 (2018)
27. Brzoska, C., Schäfer, K.: Temporal logic programming applied to image sequence evaluation. (1995)
28. Büchi, J.: Weak second-order arithmetic and finite automata. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 6(1–6), 66–92 (1960)
29. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: the *DL-Lite* family. Journal of Automated Reasoning 39(3), 385–429 (2007)
30. Chomicki, J., Toman, D.: Temporal logic in information systems. In: Chomicki, J., Saake, G. (eds.) Logics for Databases and Information Systems (the book grow out of the Dagstuhl Seminar 9529: Role of Logics in Information Systems, 1995). pp. 31–70. Kluwer (1998)
31. Chrobak, M.: Finite automata and unary languages. Theor. Comp. Sci. 47(2), 149–158 (1986)
32. Compton, K.J., Laflamme, C.: An algebra and a logic for NC^1. Inf. Comput. 87(1/2), 240–262 (1990)
33. Compton, K.J., Straubing, H.: Characterizations of regular languages in low level complexity classes. In: Paun, G., Rozenberg, G., Salomaa, A. (eds.) Current Trends in Theoretical Computer Science, Entering the 21th Century, pp. 235–246. World Scientific (2001)
34. Demri, S., Goranko, V., Lange, M.: Temporal Logics in Computer Science. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press (2016)
35. Elgot, C.: Decision problems of finite automata design and related arithmetics. Transactions of the American Mathematical Society 98, 21–51 (1961)
36. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, pp. 995–1072. Elsevier and MIT Press (1990), https://doi.org/10.1016/b978-0-444-88074-1.50021-4
37. Finkbeiner, B., Kuhtz, L.: Monitor circuits for ltl with bounded and unbounded future. In: International Workshop on Runtime Verification. pp. 60–75. Springer (2009)
38. Fisher, M.: Temporal representation and reasoning. In: van Harmelen, F., Lifschitz, V., Porter, B.W. (eds.) Handbook of Knowledge Representation, Foundations of

Artificial Intelligence, vol. 3, pp. 513–550. Elsevier (2008), https://doi.org/10.1016/S1574-6526(07)03012-X
39. Furst, M.L., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Mathematical Systems Theory 17(1), 13–27 (1984)
40. Gabbay, D., Hodkinson, I., Reynolds, M.: Temporal Logic: Mathematical Foundations and Computational Aspects, vol. 1. Oxford University Press (1994)
41. Gabbay, D., Kurucz, A., Wolter, F., Zakharyaschev, M.: Many-Dimensional Modal Logics: Theory and Applications, Studies in Logic, vol. 148. Elsevier (2003)
42. Gabelaia, D., Kontchakov, R., Kurucz, Á., Wolter, F., Zakharyaschev, M.: Combining spatial and temporal logics: Expressiveness vs. complexity. J. Artif. Intell. Res. 23, 167–243 (2005), https://doi.org/10.1613/jair.1537
43. Gabelaia, D., Kurucz, A., Wolter, F., Zakharyaschev, M.: Products of 'transitive' modal logics. J. Symb. Log. 70(3), 993–1021 (2005), https://doi.org/10.2178/jsl/1122038925
44. Glimm, B., Ogbuji, C.: SPARQL 1.1 entailment regimes. W3C Recommendation (2013), http://www.w3.org/TR/sparql11-entailment
45. Grädel, E.: On transitive closure logic. In: Börger, E., Jäger, G., Büning, H.K., Richter, M.M. (eds.) Computer Science Logic, 5th Workshop, CSL '91, Berne, Switzerland, October 7-11, 1991, Proceedings. Lecture Notes in Computer Science, vol. 626, pp. 149–163. Springer (1991)
46. Gutierrez, C., Hurtado, C.A., Vaisman, A.A.: Introducing time into RDF. IEEE Trans. Knowl. Data Eng. 19(2), 207–218 (2007)
47. Gutiérrez-Basulto, V., Jung, J.C., Ozaki, A.: On metric temporal description logics. In: Proceedings of the Twenty-second European Conference on Artificial Intelligence. pp. 837–845. IOS Press (2016)
48. Heintz, F., De Leng, D.: Spatio-temporal stream reasoning with incomplete spatial information. In: ECAI. pp. 429–434 (2014)
49. Heintz, F., Kvarnström, J., Doherty, P.: Stream reasoning in dyknow: A knowledge processing middleware system. In: Proc. 1st Intl Workshop Stream Reasoning (2009)
50. Henzinger, T.A.: Temporal specification and verification of real-time systems. Tech. rep., STANFORD UNIV CA DEPT OF COMPUTER SCIENCE (1991)
51. Hernich, A., Lutz, C., Papacchini, F., Wolter, F.: Dichotomies in ontology-mediated querying with the guarded fragment. In: Proc. of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017. pp. 185–199. ACM (2017)
52. Hirshfeld, Y., Rabinovich, A.M.: Logics for real time: Decidability and complexity. Fundam. Inform. 62(1), 1–28 (2004), http://content.iospress.com/articles/fundamenta-informaticae/fi62-1-02
53. Ho, H.M., Ouaknine, J., Worrell, J.: Online monitoring of metric temporal logic. In: International Conference on Runtime Verification. pp. 178–192. Springer (2014)
54. Hodkinson, I.M., Kontchakov, R., Kurucz, A., Wolter, F., Zakharyaschev, M.: On the computational complexity of decidable fragments of first-order linear temporal logics. In: 10th International Symposium on Temporal Representation and Reasoning / 4th International Conference on Temporal Logic (TIME-ICTL 2003), 8-10 July 2003, Cairns, Queensland, Australia. pp. 91–98. IEEE Computer Society (2003), https://doi.org/10.1109/TIME.2003.1214884
55. Hunter, P., Ouaknine, J., Worrell, J.: Expressive completeness for metric temporal logic. In: 28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013. pp. 349–357. IEEE Computer Society (2013), https://doi.org/10.1109/LICS.2013.41

56. Immerman, N.: Descriptive Complexity. Springer (1999)
57. Kamp, H.W.: Tense Logic and the Theory of Linear Order. PhD thesis, Computer Science Department, University of California at Los Angeles, USA (1968)
58. Kharlamov, E., Mehdi, G., Savkovic, O., Xiao, G., Kalayci, E.G., Roshchin, M.: Semantically-enhanced rule-based diagnostics for industrial Internet of Things: The SDRL language and case study for Siemens trains and turbines. J. Web Semant. 56, 11–29 (2019)
59. Kontchakov, R., Rodriguez-Muro, M., Zakharyaschev, M.: Ontology-based data access with databases: A short course. In: Reasoning Web. Semantic Technologies for Intelligent Data Access - 9th International Summer School 2013, Mannheim, Germany, July 30 - August 2, 2013. Proceedings. pp. 194–229 (2013), https://doi.org/10.1007/978-3-642-39784-4_5
60. Kontchakov, R., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: Boolean role inclusions in DL-Lite with and without time (2020), manuscript
61. Kontchakov, R., Zakharyaschev, M.: An introduction to description logics and query rewriting. In: Reasoning Web. Reasoning on the Web in the Big Data Era - 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings. pp. 195–244 (2014), https://doi.org/10.1007/978-3-319-10587-1_5
62. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems 2(4), 255–299 (1990), http://dx.doi.org/10.1007/BF01995674
63. de Leng, D., Heintz, F.: Approximate stream reasoning with metric temporal logic under uncertainty. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 2760–2767 (2019)
64. Libkin, L.: Elements Of Finite Model Theory. Springer (2004)
65. Lutz, C., Wolter, F., Zakharyaschev, M.: Temporal description logics: A survey. In: Proc. of the 15th Int. Symposium on Temporal Representation and Reasoning (TIME 2008). pp. 3–14 (2008)
66. Lutz, C., Sabellek, L.: Ontology-mediated querying with the description logic EL: trichotomy and linear datalog rewritability. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017. pp. 1181–1187. ijcai.org (2017), https://doi.org/10.24963/ijcai.2017/164
67. Lutz, C., Wolter, F.: The data complexity of description logic ontologies. Logical Methods in Computer Science 13(4) (2017), https://doi.org/10.23638/LMCS-13(4:7)2017
68. Manna, Z., Pnueli, A.: The temporal logic of reactive and concurrent systems - specification. Springer (1992)
69. Motik, B.: Representing and querying validity time in RDF and OWL: A logic-based approach. J. Web Semant. 12, 3–21 (2012)
70. Münch, D., IJsselmuiden, J., Arens, M., Stiefelhagen, R.: High-level situation recognition using fuzzy metric temporal logic, case studies in surveillance and smart environments. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops). pp. 882–889. IEEE (2011)
71. Ouaknine, J., Worrell, J.: On the decidability of metric temporal logic. In: 20th Annual IEEE Symposium on Logic in Computer Science (LICS'05). pp. 188–197. IEEE (2005)
72. Ouaknine, J., Worrell, J.: Some recent results in metric temporal logic. In: Cassez, F., Jard, C. (eds.) Formal Modeling and Analysis of Timed Systems, 6th International Conference, FORMATS 2008, Saint Malo, France, September 15-17, 2008.

Proceedings. Lecture Notes in Computer Science, vol. 5215, pp. 1–13. Springer (2008), https://doi.org/10.1007/978-3-540-85778-5_1
73. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. Journal on Data Semantics X, 133–173 (2008)
74. Rabinovich, A.: A proof of Kamp's theorem. Logical Methods in Computer Science 10(1) (2014)
75. Ryzhikov, V., Walega, P.A., Zakharyaschev, M.: Data complexity and rewritability of ontology-mediated queries in metric temporal logic under the event-based semantics. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. pp. 1851–1857. ijcai.org (2019), https://doi.org/10.24963/ijcai.2019/256
76. Schaerf, A.: On the complexity of the instance checking problem in concept languages with existential quantification. J. Intelligent Information Systems 2(3), 265–278 (1993)
77. Soylu, A., Giese, M., Jimenez-Ruiz, E., Vega-Gorgojo, G., Horrocks, I.: Experiencing OptiqueVQS: A multi-paradigm and ontology-based visual query system for end users. Universal Access in the Information Society 15(1), 129–152 (2016)
78. Straubing, H., Weil, P.: An introduction to finite automata and their connection to logic. CoRR abs/1011.6491 (2010)
79. Straubing, H.: Finite Automata, Formal Logic, and Circuit Complexity. Birkhauser Verlag (1994)
80. To, A.W.: Unary finite automata vs. arithmetic progressions. Inf. Process. Lett. 109(17), 1010–1014 (2009)
81. Trakhtenbrot, B.: Finite automata and the logic of one-place predicates. Siberian Mathematical Journal 3, 103–131 (1962), english translation in: AMS Transl. 59 (1966) 23–55
82. Walega, P., Cuenca Grau, B., Kaminski, M., Kostylev, E.: DatalogMTL: computational complexity and expressive power. International Joint Conferences on Artificial Intelligence (IJCAI) (2019)
83. Wałęga, P.A., Kaminski, M., Grau, B.C.: Reasoning over streaming data in metric temporal datalog. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 3092–3099 (2019)
84. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-based data access: A survey. In: Lang, J. (ed.) Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. pp. 5511–5519. ijcai.org (2018), https://doi.org/10.24963/ijcai.2018/777
85. Xiao, G., Ding, L., Cogrel, B., Calvanese, D.: Virtual knowledge graphs: An overview of systems and use cases. Data Intell. 1(3), 201–223 (2019), https://doi.org/10.1162/dint_a_00011