Ontology-Based Data Access with a Horn Fragment of Metric Temporal Logic

S. Brandt[1], **E. Güzel Kalaycı**[2], **R. Kontchakov**[3], **V. Ryzhikov**[2], **G. Xiao**[2] and **M. Zakharyaschev**[3]
[1] Siemens CT, Germany [2] Free University of Bozen-Bolzano, Italy [3] Birkbeck, University of London, UK

Abstract

We advocate *datalogMTL*, a datalog extension of a Horn fragment of the metric temporal logic *MTL*, as a language for ontology-based access to temporal log data. We show that *datalogMTL* is EXPSPACE-complete even with punctual intervals, in which case *MTL* is known to be undecidable. Nonrecursive *datalogMTL* turns out to be PSPACE-complete for combined complexity and in AC^0 for data complexity. We demonstrate by two real-world use cases that nonrecursive *datalogMTL* programs can express complex temporal concepts from typical user queries and thereby facilitate access to log data. Our experiments with Siemens turbine data and MesoWest weather data show that *datalogMTL* ontology-mediated queries are efficient and scale on large datasets of up to 11GB.

1 Introduction

Data gathering at Siemens In order to prevent malfunctions and abnormal behaviour, Siemens operates remote-diagnostic centres that gather and analyse data from installations worldwide such as gas turbines for power generation. For the service engineers working in those centres, analysing the data often begins by running queries that aggregate sensor measurements such as the power output of the turbine, its maximum rotor speed, average exhaust temperature, etc. A typical query dealing with unexpected stops of a turbine might be 'find when an active power trip occurred', that is:

(ActivePowerTrip) the active power was above 1.5MW for a period of at least 10 seconds, maximum 3 seconds after which there was a period of at least one minute where active power was below 0.15MW.

Under the traditional workflow, an engineer would call an IT expert who would produce a specific script such as

```
message("active power TRIP")  =
 $t1: eval( >, #activePower, 1.5 ) :
       for( >= 10s)
 &&
      eval( <, #activePower, 0.15 ) :
       start( after[ 0s, 3s ] $t1:end ):
        for( >= 1m);
```

for the turbine aggregated data stored in a table TB_Sensor:

turbineId	dateTime	activePower	rotorSpeed	mainFlame	...
		...			
tb0	2015-04-04 12:20:48	2	1550	0	
tb0	2015-04-04 12:20:49	1.8	1400	null	
tb0	2015-04-04 12:20:52	1.7	1350	1	
		...			

Data gathering accounts for a major part of the time service engineers require for activities at Siemens remote-diagnostic centres, most of which due to the indirect access to data. The complexity of the task stems from the lack of abstraction and the heterogeneity of data sources.

OBDA Ontology-based data access (Poggi et al. 2008) offers a different workflow that excludes the IT middleman. Domain experts develop an ontology providing definitions of the terms the engineers may be interested in together with mappings relating these terms to the database schemas. Modulo such an ontology, the query above could simply be ActivePowerTrip(tb0)@x, where x is an answer variable over time intervals. Unfortunately, the OBDA ontology and query languages standardised by W3C—the *OWL 2 QL* profile of *OWL 2* and SPARQL—are not suitable for the Siemens case as they were not designed to deal with essentially *temporal* data and concepts.

One approach to temporal OBDA is to use *OWL 2 QL* as an ontology language, assuming that ontology axioms hold at all times, and extend the query language with various temporal operators (Gutiérrez-Basulto and Klarman 2012; Baader, Borgwardt, and Lippmann 2013; Borgwardt, Lippmann, and Thost 2013; Özçep et al. 2013; Klarman and Meyer 2014; Özçep and Möller 2014; Kharlamov et al. 2016). However, *OWL 2 QL* is not able to define the temporal feature of 'active power trip', and so the engineer would have to capture it in a complex temporal query. Another known approach is to allow the temporal operators of linear temporal logic *LTL* in both queries and ontologies (Artale et al. 2013; 2015). However, sensor data come at irregular time intervals, which makes it impossible to adequately represent '10 seconds' or '1 minute' in *LTL*.

Metric temporal logic A more suitable formalism for capturing the meaning of concepts such as 'active power trip' is the logic *MTL* designed by Koymans (1990) and Alur and Henzinger (1993) for modelling and reasoning about real-time systems. *MTL* can be interpreted over the reals $(\mathbb{R}, \leq)$ and allows formulas such as $\boxminus_{[1.5,3]}\varphi$ (or $\diamondminus_{[1.5,3]}$) that hold at a moment t iff φ holds at every (respectively, some) moment in the interval $[t-3, t-1.5]$, which can easily capture the temporal feature of 'active power trip'. Unfortunately, *MTL* turns out to be undecidable (Alur and Henzinger 1993) and EXPSPACE-complete if punctual operators such as $\diamondminus_{[1,1]}$ are disallowed (Alur, Feder, and Henzinger 1996).

Our contribution In this paper, we first investigate the Horn fragment of *MTL* (without diamond operators in the head of rules) and its datalog extension *datalogMTL*, where 'active power trip' can be defined by the rule

$$\begin{aligned}&\mathsf{ActivePowerTrip}(v) \leftarrow \mathsf{Turbine}(v) \land \\ &\quad \boxminus_{[0,1m]} \mathsf{ActivePowerBelow0.15}(v) \land \\ &\quad\quad \diamondminus_{[60s,63s]} \boxminus_{[0,10s]} \mathsf{ActivePowerAbove1.5}(v), \quad (1)\end{aligned}$$

which is assumed to hold at all times. We show that answering ontology-mediated queries $(\Pi, G(\boldsymbol{v})@x)$ is EXPSPACE-complete, where Π is a *datalogMTL* program, $G(\boldsymbol{v})$ a goal with individual variables $\boldsymbol{v}$, and x a variable for intervals during which $G(\boldsymbol{v})$ holds. We also observe that *hornMTL* becomes undecidable if diamond operators are allowed in the head of rules.

From the practical point of view, most interesting are nonrecursive *datalogMTL* queries, where query answering is in AC^0 for data complexity and PSPACE-complete for combined complexity (even NP-complete if the arity of predicates is bounded). In this case, we develop a query answering algorithm that can be implemented in standard SQL (with window functions). We also present a framework for practical OBDA with nonrecursive *datalogMTL* queries and temporal log data stored in databases as above. Finally, we evaluate our framework on two use cases. We develop a *datalogMTL* ontology for temporal concepts used in typical queries at Siemens (e.g., NormalStop that takes place if events ActivePowerOff, MainFlameOff, CoastDown6600to1500, and CoastDown1500to200 happen in a certain temporal pattern). We also create a weather ontology defining standard meteorological concepts such as Hurricane (HurricaneForceWind, wind with the speed above 118 km/h, lasting at least 1 hour). Using Siemens sensor databases and MesoWest historical records of the weather stations across the US, we experimentally demonstrate that our algorithm is efficient in practice and scales on large datasets of up to 11GB.

2 Datalog*MTL*

In our applications, the intended flow of time is the real numbers $\mathbb{R}$ (but the results of this section hold also for the rational numbers $\mathbb{Q}$). By an *interval*, ι, we mean any nonempty subset of $\mathbb{R}$ of the form $[t_1, t_2]$, $[t_1, t_2)$, $(t_1, t_2]$ or (t_1, t_2) with $t_1, t_2 \in \mathbb{Q} \cup \{-\infty, \infty\}$ and $t_1 \leq t_2$ (we identify $(t, \infty]$ with (t, ∞), $[-\infty, t]$ with $(-\infty, t]$, etc.). A *range*, ϱ, is an interval with non-negative endpoints. The end-points of intervals and ranges are represented in binary. An *individual term*, τ, is an individual variable, v, or a constant, c. A *datalogMTL program*, Π, is a finite set of *rules* of the form

$$A^+ \leftarrow A_1 \land \dots \land A_k \quad \text{or} \quad \bot \leftarrow A_1 \land \dots \land A_k,$$

where $k \geq 1$, each A_i is either an inequality $\tau \neq \tau'$ or defined by the grammar

$$A ::= P(\tau_1, \dots, \tau_m) \mid \boxplus_\varrho A \mid \boxminus_\varrho A \mid \diamondplus_\varrho A \mid \diamondminus_\varrho A$$

and A^+ does not contain any diamond operators $\diamondplus_\varrho$ and $\diamondminus_\varrho$. The atoms $A_1, \dots, A_k$ constitute the *body* of the rule, while A^+ or $\bot$ its *head*. As usual, we assume that every variable in the head of a rule also occurs in its body.

A *data instance*, $\mathcal{D}$, is a finite set of *facts* of the form $P(\boldsymbol{c})@\iota$, where $P(\boldsymbol{c})$ is a ground atom and ι is an interval, stating that $P(\boldsymbol{c})$ holds throughout ι.

An *interpretation*, $\mathfrak{M}$, is based on a *domain* $\Delta \neq \emptyset$ (for the individual variables and constants). For any m-ary predicate P, m-tuple $\boldsymbol{a}$ from Δ, and moment of time $t \in \mathbb{R}$, the interpretation $\mathfrak{M}$ specifies whether P is *true on $\boldsymbol{a}$ at t*, in which case we write $\mathfrak{M}, t \models P(\boldsymbol{a})$. Let ν be an *assignment* of elements of Δ to the individual variables (we adopt the standard name assumption: $\nu(c) = c$, for every individual constant c). We then set inductively:

$$\begin{aligned}
&\mathfrak{M}, t \models^\nu P(\boldsymbol{\tau}) \quad \text{iff} \quad \mathfrak{M}, t \models P(\nu(\boldsymbol{\tau})),\\
&\mathfrak{M}, t \models^\nu \tau \neq \tau' \quad \text{iff} \quad \nu(\tau) \neq \nu(\tau'),\\
&\mathfrak{M}, t \models \boxplus_\varrho A \quad \text{iff} \quad \mathfrak{M}, s \models A \text{ for all } s \text{ with } s - t \in \varrho,\\
&\mathfrak{M}, t \models \boxminus_\varrho A \quad \text{iff} \quad \mathfrak{M}, s \models A \text{ for all } s \text{ with } t - s \in \varrho,\\
&\mathfrak{M}, t \models \diamondplus_\varrho A \quad \text{iff} \quad \mathfrak{M}, s \models A \text{ for some } s \text{ with } s - t \in \varrho,\\
&\mathfrak{M}, t \models \diamondminus_\varrho A \quad \text{iff} \quad \mathfrak{M}, s \models A \text{ for some } s \text{ with } t - s \in \varrho.
\end{aligned}$$

We say that $\mathfrak{M}$ *satisfies* Π *under* ν if, for *all* $t \in \mathbb{R}$ and all rules $A \leftarrow A_1 \land \dots \land A_k$ in Π, we have

$$\mathfrak{M}, t \models^\nu A \quad \text{whenever} \quad \mathfrak{M}, t \models^\nu A_i \quad \text{for } 1 \leq i \leq k$$

(as usual $\mathfrak{M}, t \not\models^\nu \bot$). $\mathfrak{M}$ is a *model* of Π and $\mathcal{D}$ if it satisfies Π under every assignment, and $\mathfrak{M}, t \models P(\boldsymbol{c})$ for any $P(\boldsymbol{c})@\iota$ in $\mathcal{D}$ and any $t \in \iota$. Π and $\mathcal{D}$ are *consistent* if they have a model.

Note that ranges ϱ in the temporal operators can be punctual $[t, t]$, in which case $\boxplus_{[t,t]} A$ is equivalent to $\diamondplus_{[t,t]} A$, and $\boxminus_{[t,t]} A$ to $\diamondminus_{[t,t]} A$.

A *datalogMTL query* takes the form $(\Pi, \boldsymbol{q}(\boldsymbol{v}, x))$, where $\boldsymbol{q}(\boldsymbol{v}, x) = Q(\boldsymbol{\tau})@x$, for some predicate Q, $\boldsymbol{v}$ is a tuple of individual variables occurring in the terms $\boldsymbol{\tau}$, and x an *interval variable*. A *certain answer* to $(\Pi, \boldsymbol{q}(\boldsymbol{v}, x))$ over a data instance $\mathcal{D}$ is a pair $(\boldsymbol{c}, \iota)$ such that $\boldsymbol{c}$ is a tuple of constants from $\mathcal{D}$ (of the same length as $\boldsymbol{v}$), ι an interval and, for any $t \in \iota$ and any model $\mathfrak{M}$ of Π and $\mathcal{D}$, we have $\mathfrak{M}, t \models^\nu Q(\boldsymbol{\tau})$, where ν maps $\boldsymbol{v}$ to $\boldsymbol{c}$. For example, suppose $\mathcal{D}$ consists of the facts Turbine(tb0)@$(-\infty, \infty)$, ActivePowerAbove1.5(tb0)@[13:00:00, 13:00:15), ActivePowerBelow0.15(tb0)@[13:00:17, 13:01:25), and Π is just rule (1). Then any subinterval of [13:01:17, 13:01:18) is a certain answer to $(\Pi, \mathsf{ActivePowerTrip}(\mathsf{tb0})@x)$.

By *answering datalogMTL queries* we understand the problem of checking whether a given pair $(\boldsymbol{c}, \iota)$ is a certain answer to a given *datalogMTL* query $(\Pi, \boldsymbol{q}(\boldsymbol{v}, x))$ over a given data instance $\mathcal{D}$. Our first result is the following theorem, which is to be put in the context of undecidability of *MTL* over $\mathbb{R}$, and its EXPSPACE-completeness over the integers $\mathbb{Z}$ (Alur and Henzinger 1993).

Theorem 1. *Answering datalogMTL queries is* ExpSpace-*complete* (*even in the propositional case*) *for combined complexity.*

To give the intuition behind the proof, we note first that every *datalogMTL* program Π can be transformed (using polynomially-many fresh predicates) to a *datalogMTL* program in *normal form* that contains only rules of the form

$$P(\boldsymbol{\tau}) \leftarrow \bigwedge_{i \in I} P_i(\boldsymbol{\tau}_i), \qquad \bot \leftarrow \bigwedge_{i \in I} P_i(\boldsymbol{\tau}_i), \tag{2}$$

$$\boxplus_\varrho P(\boldsymbol{\tau}) \leftarrow P'(\boldsymbol{\tau}'), \qquad \boxminus_\varrho P(\boldsymbol{\tau}) \leftarrow P'(\boldsymbol{\tau}'), \tag{3}$$

$$P(\boldsymbol{\tau}) \leftarrow \boxplus_\varrho P'(\boldsymbol{\tau}'), \qquad P(\boldsymbol{\tau}) \leftarrow \boxminus_\varrho P'(\boldsymbol{\tau}') \tag{4}$$

and gives the same certain answers as Π. For example, the rule $P(\boldsymbol{\tau}) \leftarrow P_1(\boldsymbol{\tau}_1) \land \diamondminus_\varrho P_2(\boldsymbol{\tau}_2)$ can be replaced by $P(\boldsymbol{\tau}) \leftarrow P_1(\boldsymbol{\tau}_1) \land P_2'(\boldsymbol{\tau}_2)$ and $\boxplus_\varrho P_2'(\boldsymbol{\tau}_2) \leftarrow P_2(\boldsymbol{\tau}_2)$, where P_2' is a fresh predicate of the same arity as P_2.

We now require the following notation. Given an interval ι and a range ϱ, we set

$$\iota + \varrho = \{t + k \mid t \in \iota \text{ and } k \in \varrho\}.$$

For example, if $\iota = (\iota_b, \iota_e)$ and $\varrho = [\varrho_b, \varrho_e]$, then $\iota + \varrho$ is the interval $(\iota_b + \varrho_b, \iota_e + \varrho_e)$; if $\iota = [\iota_b, \iota_e]$ and $\varrho = [\varrho_b, \varrho_e]$, then $\iota + \varrho = [\iota_b + \varrho_b, \iota_e + \varrho_e]$. Next, by $\iota - \varrho$ we denote

$$\text{the maximal interval } \iota' \text{ such that } \iota' + \varrho = \iota.$$

Note that $\iota - \varrho$ is only defined if there is t' such that $t' + k \in \iota$, for $k \in \varrho$, in which case we write $\varrho \preceq \iota$. If $\varrho \preceq \iota$, then ι' is defined uniquely. For example, if $\iota = (\iota_b, \iota_e)$ and $\varrho = [\varrho_b, \varrho_e]$ with $\varrho \preceq \iota$, then $\iota - \varrho$ is the interval $(\iota_b - \varrho_b, \iota_e - \varrho_e)$; if $\iota = (\iota_b, \iota_e)$ and $\varrho = (\varrho_b, \varrho_e)$ with $\varrho \preceq \iota$, then $\iota - \varrho = [\iota_b - \varrho_b, \iota_e - \varrho_e]$.

Now, given a data instance $\mathcal{D}$, we denote by $\Pi(\mathcal{D})$ the closure (by transfinite induction) of $\mathcal{D}$ under the rules:

(coal) if $P(\boldsymbol{c})@\iota_i \in D$, for $i \in I$, and $\bigcap_{i \in I} \iota_i \neq \emptyset$, then we add $P(\boldsymbol{c})@\bigcup_{i \in I} \iota_i$ to $\mathcal{D}$ (overlapping intervals ι_i are coalesced into their union);

(horn) if $P(\boldsymbol{c}) \leftarrow \bigwedge_{i \in I} P_i(\boldsymbol{c}_i)$ is an instance of a rule in Π with $P_i(\boldsymbol{c}_i)@\iota_i$ in $\mathcal{D}$ and $\bigcap_{i \in I} \iota_i \neq \emptyset$, then we add $P(\boldsymbol{c})@\bigcap_{i \in I} \iota_i$ to $\mathcal{D}$;

($\Box_\varrho\leftarrow$) if $\boxplus_\varrho P(\boldsymbol{c}) \leftarrow P'(\boldsymbol{c}')$ is an instance of a rule in Π with $P'(\boldsymbol{c}')@\iota \in \mathcal{D}$, then we add $P(\boldsymbol{c})@(\iota + \varrho)$ to $\mathcal{D}$ (and similarly for $\boxminus_\varrho P(\boldsymbol{c}) \leftarrow P'(\boldsymbol{c}')$);

($\leftarrow\Box_\varrho$) if $P(\boldsymbol{c}) \leftarrow \boxplus_\varrho P'(\boldsymbol{c}')$ is an instance of a rule in Π with $P'(\boldsymbol{c}')@\iota \in \mathcal{D}$ and $\varrho \preceq \iota$, then we add $P(\boldsymbol{c})@(\iota - \varrho)$ to $\mathcal{D}$ (and similarly for $P(\boldsymbol{c}) \leftarrow \boxminus_\varrho P'(\boldsymbol{c}')$).

Define a *canonical interpretation* $\mathfrak{C}_{\Pi,\mathcal{D}}$ whose object domain consists of the individual constants in Π and $\mathcal{D}$, and $\mathfrak{C}_{\Pi,\mathcal{D}}, t \models P(\boldsymbol{c})$ iff $P(\boldsymbol{c})@\iota \in \Pi(\mathcal{D})$, for some $\iota \ni t$.

Lemma 2. (*i*) *If* $\bot@\iota \in \Pi(\mathcal{D})$ *for some* ι, *then* Π *and* $\mathcal{D}$ *are inconsistent*; *otherwise,* $\mathfrak{C}_{\Pi,\mathcal{D}}$ *is the minimal model of* Π *and* $\mathcal{D}$ *in the sense that* $P(\boldsymbol{c})@\iota \in \Pi(\mathcal{D})$ *implies* $\mathfrak{M}, t \models P(\boldsymbol{c})$, *for any model* $\mathfrak{M}$ *of* Π *and* $\mathcal{D}$ *and any* $t \in \iota$.

(*ii*) *A pair* $(\boldsymbol{c}, \iota)$ *is a certain answer to* $(\Pi, \boldsymbol{q}(\boldsymbol{v}, x))$ *over* $\mathcal{D}$ *consistent with* Π *iff* $\mathfrak{C}_{\Pi,\mathcal{D}}, t \models \boldsymbol{q}(\boldsymbol{c})$ *for all* $t \in \iota$.

Let $\mathbf{1}$ be the greatest common divisor of the (rational) numbers in Π and $\mathcal{D}$. Let $\mathsf{grid}(\Pi, \mathcal{D})$ be the closure of these numbers under the operations $+\mathbf{1}$ and $-\mathbf{1}$. It is not hard to see that the order $(\mathsf{grid}(\Pi, \mathcal{D}), \leq)$ is isomorphic to $(\mathbb{Z}, \leq)$.

Lemma 3. *For any ground* $P(\boldsymbol{c})$ *and any* $t \in \mathsf{grid}(\Pi, \mathcal{D})$, *we either have* $\mathfrak{C}_{\Pi,\mathcal{D}}, t' \models P(\boldsymbol{c})$, *for all* $t' \in (t, t + \mathbf{1})$, *or* $\mathfrak{C}_{\Pi,\mathcal{D}}, t' \not\models P(\boldsymbol{c})$, *for all* $t' \in (t, t + \mathbf{1})$.

The ExpSpace upper bound in Theorem 1 can now be obtained by (exponential) reduction to *LTL* over $\mathbb{Z}$, which is known to be PSpace-complete (Sistla and Clarke 1985).

The diamond operators $\diamondplus_\varrho$ and $\diamondminus_\varrho$ are disallowed in the head of *datalogMTL* rules for the following reason. Denote by *datalogMTL*$^\diamond$ the extension of *datalogMTL* that allows arbitrary temporal operators in the head of rules. The extended language turns out to be much more powerful and can encode 2-counter Minsky machines, which gives the following theorem; cf. (Madnani, Krishna, and Pandya 2013).

Theorem 4. *Answering datalogMTL*$^\diamond$ *queries is undecidable.*

As none of the *datalogMTL* programs required in our use cases is recursive, we now consider the class *datalog*$_{nr}$*MTL* of nonrecursive *datalogMTL* programs. More precisely, for a program Π, let $\lessdot$ be the dependence relation on the predicate symbols in Π: $P \lessdot Q$ iff Π has a clause with P in the head and Q in the body. Π is called *nonrecursive* if $P \lessdot^* P$ does not hold for any predicate symbol P in Π, where $\lessdot^*$ is the transitive closure of $\lessdot$.

For *datalog*$_{nr}$*MTL* queries, one can define a *finite* order $\mathsf{grid}(\Pi, \mathcal{D})$ of exponential size, for which Lemma 3 holds with two additional infinite intervals $(-\infty, \min)$ and $(\max, \infty)$, where $\max$ and $\min$ are the maximal and minimal integers occurring in $\mathcal{D}$; if they do not exist, $\mathsf{grid}(\Pi, \mathcal{D})$ is just one interval $(-\infty, \infty)$. Since $\mathsf{grid}(\Pi, \mathcal{D})$ can be encoded in polynomial space, we can use a tableau-like top-down procedure to obtain a PSpace upper bound for answering nonrecursive *datalogMTL* queries:

Theorem 5. *Answering datalog*$_{nr}$*MTL queries is* PSpace-*complete for combined complexity* (*even in the propositional case*) *and in* AC^0 *for data complexity.*

The following example shows how a *datalog*$_{nr}$*MTL* program can generate all possible assignments of truth-values to given propositional variables, which is required in the proof of the PSpace lower bound in Theorem 5.

Example 6. Let Π_3 be a *datalog*$_{nr}$*MTL* program with propositional variables $p, q, r, \dots$ as well as $\bar{p}, \bar{q}, \bar{r}, \dots$ (representing $\neg p, \neg q, \neg r, \dots$) and the rules

$$r^\sigma \leftarrow r_0^\sigma, \tag{5}$$

$$q^\sigma \leftarrow q_0^\sigma, \qquad q^\sigma \leftarrow \diamondminus_{[4,4]} q_0^\sigma, \tag{6}$$

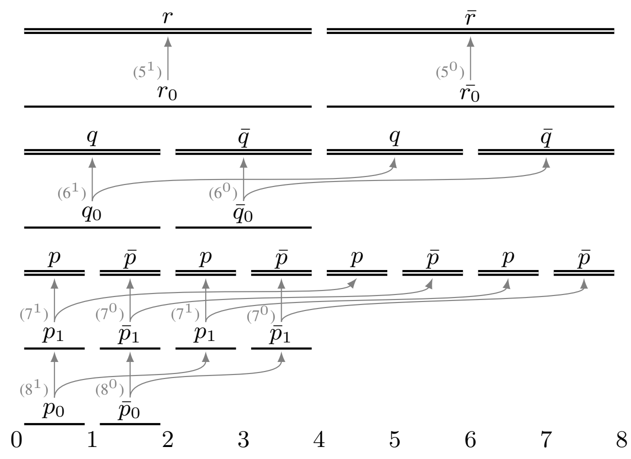
$$p^\sigma \leftarrow p_1^\sigma, \qquad p^\sigma \leftarrow \diamondminus_{[4,4]} p_1^\sigma, \tag{7}$$

$$p_1^\sigma \leftarrow p_0^\sigma, \qquad p_1^\sigma \leftarrow \diamondminus_{[2,2]} p_0^\sigma, \tag{8}$$

where $\sigma \in \{0, 1\}$, $s^1 = s$ and $s^0 = \bar{s}$, for any propositional variable s. Let $\mathcal{D}$ be a data instance with the facts

$$p_0@[0,1), \quad \bar{p}_0@[1,2), \quad q_0@[0,2), \quad \bar{q}_0@[2,4), \quad r_0@[0,4), \quad \bar{r}_0@[4,8).$$

The canonical model $\mathfrak{C}_{\Pi_3,\mathcal{D}}$ is shown below, where an arrow (i^σ) indicates an application of the rule (i) in Π_3 for σ:



Note that the Horn fragment of the Halpern-Shoham logic $\mathcal{HS}$ is P-complete over dense orders but undecidable over discrete ones (Bresolin et al. 2016), while the Horn fragment of *LTL* is P-complete without the next operator, PSPACE-complete with the next operator, and P-complete in the non-recursive case (even with next) over $\mathbb{Z}$ (Artale et al. 2014).

3 Implementing *datalog$_{nr}$MTL* in SQL

In our applications, instead of the PSPACE top-down tableau procedure we use a rewriting approach that produces an SQL query implementing a bottom-up evaluation. Namely, we rewrite a given *datalog$_{nr}$MTL* query $(\Pi, Q(\boldsymbol{\tau})@x)$ with Π in normal form (2)–(4) to an SQL query computing the certain answers $(\boldsymbol{c}, \iota)$ to the query with *maximal* intervals ι.

In a nutshell, the rewriting algorithm produces SQL views that apply the rules (coal), (horn), $(\boxminus_\varrho\leftarrow)$ and $(\leftarrow\boxminus_\varrho)$ above to the facts extracted from the database using mappings. The algorithm starts with tables P containing these facts and having all the temporal intervals sorted (which is usually the case for log data and mappings such as $\mathcal{M}$ in Section 4). Each table P is coalesced by the algorithm from (Zhou, Wang, and Zaniolo 2006) into a new table in time $O(|P|)$. By applying a rule $(\boxminus_\varrho\leftarrow)$ or $(\leftarrow\boxminus_\varrho)$ to P', we construct a table P in time $O(|P'|)$. The rule (horn) is applied to the tables P_i, for $i \in I$, by a variant of the merge join algorithm in time $O(|I|n^{|I|}m)$, where n is the maximum number of individual tuples in P_i, $i \in I$, and m is the maximum number of interval tuples. We then form the union of the k-many tables constructed for the same P; as the intervals in them are sorted, we obtain a sorted table for P in time $O(|I|n^{|I|}mk)$ and then coalesce it in linear time. Observe that the time required to compute the resulting table P is polynomial in n (of degree $|I|$) and linear in m, which explains linear patterns in our experiments below, where the size of individual tuples is fixed. To compute the table for the goal Q, we iterate the described procedure d times, where d is the length of the longest chain of predicates in the dependence relation $\lessdot$ for Π. Thus, the overall time required to compute the goal predicate Q is exponential in d and the size of Π itself, polynomial in n and linear in m.

4 Use Cases

We test the feasibility of OBDA with *datalog$_{nr}$MTL* by querying Siemens turbine log data and MesoWest weather data. First, we briefly describe these use cases.

Siemens service centres store aggregated turbine sensor data in tables such as TB_Sensor. The data comes with (not necessarily regular) timestamps $t_1, t_2, \dots$, and it is deemed that the values remain constant in every interval $[t_i, t_{i+1})$. Using a set of mappings, we extract from these tables a data instance containing ground facts such as

$$\begin{array}{l}\mathsf{ActivePowerAbove1.5(tb0)@[12{:}20{:}48, 12{:}20{:}49)},\\ \mathsf{ActivePowerAbove1.5(tb0)@[12{:}20{:}49, 12{:}20{:}52)},\\ \mathsf{RotorSpeedAbove1500(tb0)@[12{:}20{:}48, 12{:}20{:}49)},\\ \mathsf{MainFlameBelow0.1(tb0)@[12{:}20{:}48, 12{:}20{:}52)}.\end{array}$$

For example, the first two of them are obtained from the table TB_Sensor using the following SQL mapping $\mathcal{M}$:

```
ActivePowerAbove1.5(x)@[t1, t2) ←
  SELECT x, t1, t2 FROM (
   SELECT turbineId AS x,
     LAG(dateTime) OVER (w) AS t1,
     LAG(activePower) OVER (w) AS lag_activePower,
     dateTime AS t2
   FROM TB_Sensor
   WINDOW w AS (PARTITION BY turbineId
     ORDER BY dateTime)
   ) tmp
  WHERE lag_activePower > 1.5
```

In terms of the basic predicates above, we define more complex ones that are used in queries posed by the Siemens engineers. Below is a snippet of our *datalog$_{nr}$MTL* ontology (the complete ontology can be found in the technical report (Brandt et al. 2016)):

$$\begin{array}{l}\mathsf{NormalStop}(v) \leftarrow \mathsf{CoastDown1500to200}(v) \land \\ \qquad \diamondminus_{(0,9m]}\big[\mathsf{CoastDown6600to1500}(v) \land \\ \qquad\qquad \diamondminus_{(0,2m]}\big(\mathsf{MainFlameOff}(v) \land \\ \qquad\qquad\qquad \diamondminus_{(0,2m]}\mathsf{ActivePowerOff}(v)\big)\big],\\ \mathsf{MainFlameOff}(v) \leftarrow \boxminus_{[0s,10s]}\mathsf{MainFlameBelow0.1}(v),\\ \mathsf{ActivePowerOff}(v) \leftarrow \boxminus_{[0s,10s]}\mathsf{MainPowerBelow0.15}(v),\\ \mathsf{CoastDown6600to1500}(v) \leftarrow \\ \qquad \boxminus_{[0s,30s]}\,\mathsf{RotorSpeedBelow1500}(v) \land \\ \qquad\qquad \diamondminus_{(0,2m]}\,\boxminus_{(0,30s]}\,\mathsf{RotorSpeedAbove6600}(v),\\ \mathsf{CoastDown1500to200}(v) \leftarrow \\ \qquad \boxminus_{[0s,30s]}\,\mathsf{RotorSpeedBelow200}(v) \land \\ \qquad\qquad \diamondminus_{(0,9m]}\,\boxminus_{(0,30s]}\,\mathsf{RotorSpeedAbove1500}(v),\\ \mathsf{NormalRestart}(v) \leftarrow \\ \qquad\qquad \mathsf{NormalStart}(v) \land \diamondminus_{(0,1h]}\mathsf{NormalStop}(v).\end{array}$$

MesoWest. The MesoWest[1] project makes publicly available historical records of the weather stations across the US showing such parameters of meteorological conditions as temperature, wind speed and direction, amount of precipitation, etc. Each station outputs its measurements with some periodicity, with the output at a time t_{i+1} containing the accumulative (e.g., for precipitation) or averaged (e.g., for wind speed) value over the interval $(t_i, t_{i+1}]$. The data comes in a table Weather, which looks as follows:

stationId	dateTime	airTemp	windSpeed	windDir	hourPrecip	…
		…				
KBVY	2013-02-15;15:14	8	45	10	0.05	
KMNI	2013-02-15;15:21	6	123	240	0	
KBVY	2013-02-15;15:24	8	47	10	0.08	
KMNI	2013-02-15;15:31	6.7	119	220	0	
		…				

One more table, Metadata, provides some atemporal meta information about the stations:

stationId	county	state	latitude	longitude	…
		…			
KBVY	Essex	Massachusetts	42.58361	-70.91639	
KMNI	Essex	Massachusetts	33.58333	-80.21667	
		…			

The monitoring and historical analysis of the weather involves answering queries such as 'find showery counties, where one station observes precipitation at the moment, while another one does not, but observed precipitation 30 minutes ago'.

We use SQL mappings over the Weather table similar to those in the Siemens case to obtain ground atoms such as

$$\begin{aligned}&\mathsf{NorthWind}(\mathsf{KBVY})@(15{:}14, 15{:}24],\\ &\mathsf{HurricaneForceWind}(\mathsf{KMNI})@(15{:}21, 15{:}31],\\ &\mathsf{Precipitation}(\mathsf{KBVY})@(15{:}14, 15{:}24],\\ &\mathsf{TempAbove0}(\mathsf{KBVY})@(15{:}14, 15{:}24],\\ &\mathsf{TempAbove0}(\mathsf{KMNI})@(15{:}21, 15{:}31]\end{aligned}$$

(according to the standard definition, the hurricane force wind is above 118 km/h). On the other hand, mappings to the Metadata table provide atoms such as

$$\begin{aligned}&\mathsf{LocatedInCounty}(\mathsf{KBVY}, \mathsf{Essex})@(-\infty, \infty),\\ &\mathsf{LocatedInState}(\mathsf{KBVY}, \mathsf{Massachusetts})@(-\infty, \infty).\end{aligned}$$

Our ontology contains definitions of various meteorological terms; a few examples are given below:

$$\begin{aligned}&\mathsf{ShoweryCounty}(v) \leftarrow \mathsf{LocatedInCounty}(u_1, v) \land\\ &\quad \mathsf{LocatedInCounty}(u_2, v) \land \mathsf{Precipitation}(u_1) \land\\ &\qquad \mathsf{NoPrecipitation}(u_2) \land \diamondminus_{(0,30m]}\mathsf{Precipitation}(u_2),\\ &\boxminus_{[0,1h]}\ \mathsf{Hurricane}(v) \leftarrow \boxminus_{[0,1h]}\mathsf{HurricaneForceWind}(v),\\ &\mathsf{HurricaneAffectedState}(v) \leftarrow \mathsf{LocatedInState}(u, v) \land\\ &\qquad\qquad \mathsf{Hurricane}(u),\\ &\boxminus_{[0,24h]}\ \mathsf{ExcessiveHeat}(v) \leftarrow \boxminus_{[0,24h]}\mathsf{TempAbove24}(v) \land\\ &\qquad\qquad \diamondminus_{[0,24h]}\mathsf{TempAbove41}(v),\end{aligned}$$

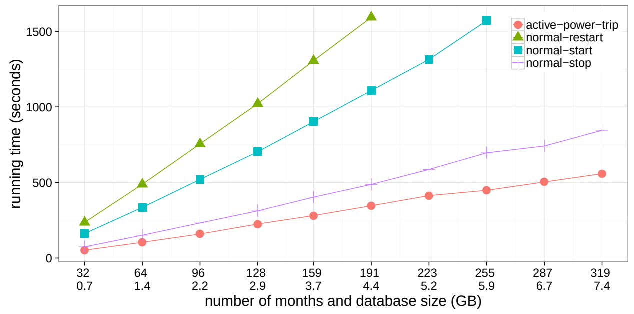
[1]`http://mesowest.utah.edu/`

$$\begin{aligned}&\mathsf{HeatAffectedCounty}(v) \leftarrow \mathsf{LocatedInCounty}(u, v) \land\\ &\qquad\qquad \mathsf{ExcessiveHeat}(u),\\ &\mathsf{CyclonePatternState}(v) \leftarrow \mathsf{LocatedInState}(u_1, v) \land\\ &\quad \mathsf{LocatedInState}(u_2, v) \land \mathsf{LocatedInState}(u_3, v) \land\\ &\qquad \mathsf{LocatedInState}(u_4, v) \land \mathsf{EastWind}(u_1) \land\\ &\qquad\quad \mathsf{NorthWind}(u_2) \land \mathsf{WestWind}(u_3) \land \mathsf{SouthWind}(u_4).\end{aligned}$$

5 Experiments

To evaluate the performance of the SQL queries produced by the *datalog$_{nr}$MTL* rewriting algorithm outlined above, we developed two benchmarks for our use cases. We ran the experiments on an HP Proliant server with 24 Intel Xeon CPUs (@3.47GHz), 106GB of RAM and five 1TB 15K RPM HD. We used PostgreSQL as a database engine. The maximum physical memory consumption in our experiments was 12.9GB.

Siemens provided us with a sample of data for one running turbine, which we denote by tb0, over 4 days in the form of the table TB_Sensor. The data table was rather sparse, containing a lot of nulls, because different sensors recorded data at different frequencies. For example, ActivePower arrived most frequently with average periodicity of 7 seconds, whereas the values for the field MainFlame arrived most rarely, every 1 minute on average. We replicated this sample to imitate the data for one turbine over 10 different periods ranging from 32 to 320 months. The statistics of the data sets is given in Table 1a. With a timeout of 30 minutes, we evaluated four queries $\mathsf{ActivePowerTrip}(\mathsf{tb0})@x$, $\mathsf{NormalStart}(\mathsf{tb0})@x$, $\mathsf{NormalStop}(\mathsf{tb0})@x$, and $\mathsf{NormalRestart}(\mathsf{tb0})@x$. The execution times are given in the picture below, which shows their linear growth in the number of months and, consequently, in the size of data (confirming theoretical results since we deal with a single turbine).



Note that the normal restart (start) query timeouts on the data for more than 15 (respectively, 20) years, which is more than enough for the monitoring and diagnostics tasks at Siemens, where the two most common application scenarios for sensor data analytics are daily monitoring (that is, analytics of high-frequency data of the previous 24 hours) and fleet-level analytics of key-performance indicators over one year. In both cases, the computation time of the results is far less a crucial cost factor than the lead-time for data preparation.

MesoWest. In contrast to the Siemens case, the weather tables contain very few nulls. Normally, the data values arrive with periodicity from 1 to 20 minutes. We tested

# months	32	64	96	128	159	191	223	255	287	320
raw size (GB)	0.7	1.4	2.2	2.9	3.7	4.4	5.2	5.9	6.7	7.4
total size (GB)	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0

(a) Siemens data for one turbine.

# years	1	2	3	4	5	6	7	8	9	10
raw size (GB)	0.3	0.8	1.4	2.0	2.7	3.7	4.9	6.1	7.7	11.0
total size (GB)	0.4	1.1	2.0	2.9	3.9	5.4	7.1	8.9	11.0	14.0

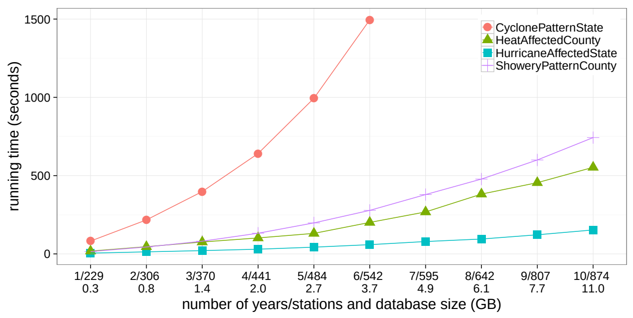
(b) NY weather stations from 2005 to 2014.

# states	2	3	4	6	8	10	12	14	17	19
names	DE,GA	+NY	+MD	+NJ,RI	+MA,CT	+LA,VT	+ME,WV	+NH,NC	+MS,SC,ND	+KY,SD
raw size (GB)	1.2	2.4	3.1	3.9	5.1	6.1	7.1	8.1	9.2	10.0
total size (GB)	2.0	4.1	5.3	6.5	8.6	10.0	12.0	14.0	16.0	18.0

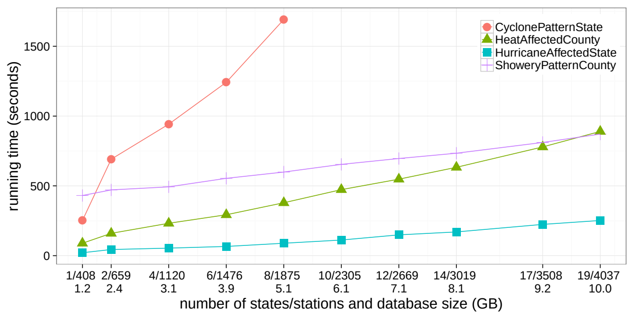
(c) Weather data for 1–19 states in 2012.

Table 1: The size of the data sets used in the experiments.
Raw size: the size of the data itself stored in PostgreSQL reported by the `pg_relation_size` function.
Total size: the size of the total data (including the index) stored in PostgreSQL reported by the `pg_total_relation_size` function.

the performance of our algorithm by increasing (i) the temporal span (with some necessary increase of the spatial spread) and (ii) the geographical spread of data. For (i), we took the New York state data for the 10 continuous periods between 2005 and 2014; see Table 1b. As each year around 70 new weather stations were added, our 10 data samples increase more than linearly in size. For (ii), we fixed the time period of one year (2012) and linearly increased the data from 1 to 19 states (NY, NJ, MD, DE, GA, RI, MA, CT, LA, VT, ME, WV, NH, NC, MS, SC, ND, KY, SD); see Table 1c. In both cases, with a timeout of 30 minutes, we executed four *datalog*$_{nr}$*MTL* queries $\mathsf{ShoweryCounty}(v)@x$, $\mathsf{HurricaneAffectedState}(\mathsf{NY})@x$, $\mathsf{HeatAffectedCounty}(v)@x$, $\mathsf{CyclonePatternState}(\mathsf{NY})@x$. The execution times are given in the pictures below (depending on the number of years in the first case and the number of states in the second):



In the experiment (i), we observe a mild non-linear dependency (the growth is even closer to linear if measured in the size of the data sets, as 70 new stations are added each year on average). The experiment (ii) exhibits linear behaviour even though about 500 stations are added in each new data set. The linear behaviour in that case can be explained by the fact that the data can be naturally partitioned into 'chunks' according to the location states of the stations so that the chunks are independent in the sense that, for any query Q and chunks $\mathcal{D}$ and $\mathcal{D}'$, we have $Q(\mathcal{D} \cup \mathcal{D}') = Q(\mathcal{D}) \cup Q(\mathcal{D}')$. Such linear performance is then realised by PostgreSQL taking advantage of proper indexes over individuals and intervals. Note that the cyclone pattern state query is most expensive because its definition includes a join of four atoms for winds in four directions, each with a large volume of instances.



Overall, the results of the experiments look very encouraging: our *datalog*$_{nr}$*MTL* query rewriting algorithm produces SQL queries that are executable by a standard database engine PostgreSQL in acceptable time over large sets of real-world temporal data of up to 11GB. The relatively challenging queries such as $\mathsf{NormalRestart}$ and $\mathsf{CyclonePatternState}$ require a large number of temporal joins, which turn out to be rather expensive. One promising optimisation could be to employ distributed computing techniques to further exploit spatial/temporal partitions.

6 Conclusions and Future Work

To facilitate access to sensor temporal data with the aim of monitoring and diagnostics, we suggested the ontology language *datalogMTL*, a combination of datalog with the Horn fragment of the metric temporal logic *MTL*. We showed that answering *datalogMTL* queries is EXPSPACE-complete for combined complexity, but becomes undecidable if the diamond operators are allowed in the head of rules. We also proved that answering nonrecursive *datalogMTL* queries is PSPACE-complete for combined complexity and in AC^0 for data complexity. We tested feasibility and efficiency of OBDA with *datalog*$_{nr}$*MTL* on two real-world use cases by querying Siemens turbine data and MesoWest weather data. Namely, we designed $datalog_{nr}MTL$ ontologies defining typical concepts used by Siemens engineers and various meteorological terms, developed and implemented an algorithm rewriting $datalog_{nr}MTL$ queries into SQL queries, and then executed the SQL queries obtained by this algorithm from our ontologies over the Siemens and MesoWest data, showing their acceptable efficiency and scalability. (To the best of our knowledge, this is the first work on practical OBDA with temporal ontologies, and so no other systems with similar functionalities are available for comparison.)

Based on these encouraging results, we plan to extend *datalogMTL* with the *since* and *until* operators and include our temporal OBDA framework into the Ontop platform (Rodriguez-Muro, Kontchakov, and Zakharyaschev 2013; Kontchakov et al. 2014; Calvanese et al. 2017).[2] We are also working on the streaming data setting, where the challenge is to continuously evaluate queries over the incoming data.

Acknowledgements

This paper was supported by the EU IP Optique, n. FP7-318338, and the UK EPSRC project iTract EP/M012670. We are grateful to Diego Calvanese for helpful discussions.

References

Alur, R., and Henzinger, T. A. 1993. Real-time logics: Complexity and expressiveness. *Inf. Comput.* 104(1):35–77.

Alur, R.; Feder, T.; and Henzinger, T. A. 1996. The benefits of relaxing punctuality. *J. ACM* 43(1):116–146.

Artale, A.; Kontchakov, R.; Wolter, F.; and Zakharyaschev, M. 2013. Temporal description logic for ontology-based data access. In *Proc. of IJCAI*, 711–717. IJCAI/AAAI.

Artale, A.; Kontchakov, R.; Ryzhikov, V.; and Zakharyaschev, M. 2014. A cookbook for temporal conceptual data modelling with description logics. *ACM Trans. Comput. Log.* 15(3):25:1–25:50.

Artale, A.; Kontchakov, R.; Kovtunova, A.; Ryzhikov, V.; Wolter, F.; and Zakharyaschev, M. 2015. First-order rewritability of temporal ontology-mediated queries. In *Proc. of IJCAI*, 2706–2712. IJCAI/AAAI.

Baader, F.; Borgwardt, S.; and Lippmann, M. 2013. Temporalizing ontology-based data access. In *Proc. of the 24th Int. Conf. on Automated Deduction, CADE-24*, volume 7898 of *LNCS*, 330–344. Springer.

Borgwardt, S.; Lippmann, M.; and Thost, V. 2013. Temporal query answering in the description logic DL-Lite. In *Proc. of the 9th Int. Symposium on Frontiers of Combining Systems, FroCoS'13*, volume 8152 of *LNCS*, 165–180. Springer.

Brandt, S.; Güzel Kalaycı, E.; Kontchakov, R.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2016. Ontology-based data access with a Horn fragment of metric temporal logic (technical report). http://www.inf.unibz.it/~gxiao/AAAI-17-MTL-Datalog-TR.pdf.

Bresolin, D.; Kurucz, A.; Muñoz-Velasco, E.; Ryzhikov, V.; Sciavicco, G.; and Zakharyaschev, M. 2016. Horn fragments of the Halpern-Shoham interval temporal logic (technical report). *CoRR* abs/1604.03515.

Calvanese, D.; Cogrel, B.; Komla-Ebri, S.; Kontchakov, R.; Lanti, D.; Rezk, M.; Rodríguez-Muro, M.; and Xiao, G. 2017. Ontop: Answering SPARQL queries over relational databases. *Semantic Web Journal*.

Gutiérrez-Basulto, V., and Klarman, S. 2012. Towards a unifying approach to representing and querying temporal data in description logics. In *Proc. of RR*, volume 7497 of *LNCS*, 90–105. Springer.

Kharlamov, E.; Brandt, S.; Jiménez-Ruiz, E.; Kotidis, Y.; Lamparter, S.; Mailis, T.; Neuenstadt, C.; Özçep, Ö. L.; Pinkel, C.; Svingos, C.; Zheleznyakov, D.; Horrocks, I.; Ioannidis, Y. E.; and Möller, R. 2016. Ontology-based integration of streaming and static relational data with optique. In *Proc. of the 2016 Int. Conf. on Management of Data, SIGMOD Conference 2016*, 2109–2112. ACM.

Klarman, S., and Meyer, T. 2014. Querying temporal databases via OWL 2 QL. In *Proc. of the 8th Int. Conf. on Web Reasoning and Rule Systems, RR 2014*, volume 8741 of *LNCS*, 92–107. Springer.

Kontchakov, R.; Rezk, M.; Rodriguez-Muro, M.; Xiao, G.; and Zakharyaschev, M. 2014. Answering SPARQL queries over databases under OWL 2 QL entailment regime. In *Proc. of ISWC, Part I*, volume 8796 of *LNCS*, 552–567. Springer.

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

Madnani, K.; Krishna, S. N.; and Pandya, P. K. 2013. On the decidability and complexity of some fragments of Metric Temporal Logic. *CoRR* abs/1305.6137.

Özçep, Ö., and Möller, R. 2014. Ontology based data access on temporal and streaming data. In *the 10th Int. Summer School on Reasoning Web, RW 2014*, volume 8714 of *LNCS*, 279–312. Springer.

Özçep, Ö.; Möller, R.; Neuenstadt, C.; Zheleznyakov, D.; and Kharlamov, E. 2013. A semantics for temporal and stream-based query answering in an OBDA context. Technical report, Deliverable D5.1, FP7-318338, EU.

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

Rodríguez-Muro, M.; Kontchakov, R.; and Zakharyaschev, M. 2013. Ontology-based data access: Ontop of databases. In *Proc. of ISWC, Part I*, volume 8218 of *LNCS*, 558–573. Springer.

Sistla, A., and Clarke, E. 1985. The complexity of propositional linear temporal logics. *J. ACM* 32:733–749.

Zhou, X.; Wang, F.; and Zaniolo, C. 2006. Efficient temporal coalescing query support in relational database systems. In *Proc. of the 17th Int. Conf. on Database and Expert Systems Applications, DEXA 2006*, volume 4080 of *LNCS*, 676–686. Springer.

[2] `http://ontop.inf.unibz.it/`