Querying Temporal Databases via OWL 2 QL

Szymon Klarman and Thomas Meyer

Centre for Artificial Intelligence Research,
CSIR Meraka and University of KwaZulu-Natal, South Africa
{sklarman,tmeyer}@csir.co.za

Abstract. SQL:2011, the most recently adopted version of the SQL query language, has unprecedentedly standardized the representation of temporal data in relational databases. Following the successful paradigm of ontology-based data access, we develop a practical approach to querying the SQL:2011-based temporal data model via the semantic layer of OWL 2 QL. The interval-based *temporal query language* (TQL), which we propose for this task, is based on naturally characterizable combinations of temporal logic with conjunctive queries. As the central contribution, we present rules for sound and complete rewriting of TQL queries into two-sorted first-order logic, and consequently, into corresponding SQL queries, which can be evaluated in any existing relational database management system compliant with the SQL:2011 temporal data model. Importantly, the proposed rewriting is based on the direct reuse of the standard rewriting techniques for conjunctive queries under OWL 2 QL. This renders our approach modular and easily implementable. As a notable corollary, we show that the data complexity of TQL query answering remains in AC^0, i.e., as in the usual, non-temporal case.

1 Introduction

The ability to manage the temporal aspects of information is critical for a variety of applications. One natural and prevailing scenario is that of representing and querying the *validity time* of data, i.e., the time during which data is deemed true about the application domain. The significance of this task is particularly visible in the area of semantic technologies, where the systematically growing number of proposed solutions, building on different levels of the Semantic Web architecture and differing in the flavour and depth of temporal reasoning they support, aim at addressing essentially the same problem [15,14,5,22,7,2]. A very similar proliferation of proposals was witnessed in the 1990s in the field of temporal databases. Intensive attempts to extend the traditional relational data model and SQL with temporal features inspired then a large body of candidate specifications, including such extensions as TSQL2, SQL3 or SQL/Temporal [24], which eventually failed to be adopted by the database community due to the persistent lack of consensus as to the preferred approach. Only very recently, that discussion has been picked up again and a compromise temporal extension has eventually found its way into SQL:2011 [20] — the newest standardization of the SQL query language. This unprecedented circumstance offers an interesting opportunity to address the problem of reasoning with temporal semantic data from yet another angle, namely, by relating it via known links between relational databases and semantic technologies to its analogue in the database world, thus using the SQL:2011 standard as a leverage for the solution. In this paper, we contribute precisely to this research agenda by proposing a novel, temporal extension to the framework of ontology-based data access.

Ontology-based data access (OBDA) is a popular paradigm of managing information, which combines the data storage and querying capabilities offered by relational database management systems (RDBMSs) with the semantically enhanced view on the data provided by ontologies. The ontology language OWL 2 QL, based on the DL-*Lite* family of Description Logics, is a profile OWL 2 designed specifically to support optimally balanced OBDA. In a nutshell, conjunctive queries, posed over data under an OWL 2 QL ontology, can be rewritten into first-order logic using the ontology's axioms, then translated to SQL and answered within an RDBMS, in such a way as if the ontology was mediating in the process [8]. As large portions of data available through the Web are in fact still hosted in relational datastores, OBDA provides a crucial channel for accessing this data from the level of the Semantic Web applications.

In this work, we establish an analogical OBDA interface between the semantic layer of OWL 2 QL and temporal data model endorsed by SQL:2011. The interval-based *temporal query language* (TQL), which we propose for this task, is based on naturally characterizable combinations of temporal logic with conjunctive queries, identified in [16]. TQL is tailored specifically to offer maximum expressivity while preserving the possibility of reuse central to OBDA first-order rewriting techniques and tools, developed specifically for the use with OWL 2 QL. While this technical compliance warrants the minimal implementation overhead for our approach, its well-defined logic foundations allow us to identify basic formal properties of TQL. In particular, we are able to demonstrate that under the finite time domain assumption the data complexity of query entailment remains in AC^0, i.e., as in the case of standard (non-temporal) conjunctive queries, even though the combined complexity increases to PSpace-complete. As the main contribution, we develop a rewriting of TQL queries in the presence of OWL 2 QL ontologies into two-sorted first-order logic, and consequently to SQL, which opens the way to efficient query answering by means of existing, commercially supported RDBMSs, such as IBM DB2 10.1, Oracle Database 11g Workspace Manager, or Teradata — all of which have by now adopted certain variants of the SQL:2011 standard[1].

The paper is organized as follows. In the next section we lay down the preliminaries. In Section 3, we introduce TQL and define the query entailment problem. In Section 4, we present the TQL query rewriting rules and, in Section 5, we study the formal properties of query answering queries via this rewriting. The related work is discussed in Section 6 and the paper is concluded in Section 7. Some proofs are included in the online technical report [19].

[1] See `http://www.cs.arizona.edu/~rts/sql3.html` for an overview.

2 Basic Notions

We start by recapping the logic foundations of OBDA. Then we motivate and formally introduce the temporal extension of OBDA studied in this paper.

2.1 Ontology-Based Data Access

OWL 2 QL is a profile of OWL 2 based on the DL-*Lite* family of Description Logics (DLs) [8]. A DL vocabulary $\Sigma = (\mathsf{N_I}, \mathsf{N_C}, \mathsf{N_R})$ consists of countably infinite sets of individual names ($\mathsf{N_I}$), concept names ($\mathsf{N_C}$) and role names ($\mathsf{N_R}$). An ABox $\mathcal{A}$ is a finite set of assertions of type $A(a)$ and $r(a, b)$, for $a, b \in \mathsf{N_I}$, $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$, which we also generically denote with $\alpha(\boldsymbol{a})$. A TBox $\mathcal{T}$ is a finite set of concept inclusions $B \sqsubseteq C$ and role inclusions $r \sqsubseteq s$, where B, C and r, s are possibly complex concepts and roles, respectively, built using logical constructors allowed in OWL 2 QL, such as $\exists r.\top, A \sqcap B, r^-$, whose particulars are not of importance for this work.[2] The semantics is given in terms of DL interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, defined as usual [4]. An interpretation $\mathcal{I}$ is a model of $\mathcal{T}$ and $\mathcal{A}$, denoted as $\mathcal{I} \models \mathcal{T}, \mathcal{A}$, *iff* it satisfies every axiom in $\mathcal{T}$ and $\mathcal{A}$.

In OBDA, the instance data, represented as an ABox, is accessed via an ontology, given as a TBox, using a designated query language such as, most commonly considered in that context, the language of conjunctive queries [13]. Let $\mathsf{N_V}$ be a countably infinite set of variables. A *conjunctive query* (CQ) over a DL vocabulary Σ is a first-order formula:

$$q(\boldsymbol{y}) = \exists \boldsymbol{x}.(\bigwedge_{1 \leq j \leq n} \alpha_j(\boldsymbol{d}_j))$$

where $\boldsymbol{y}$ and $\boldsymbol{x}$ are sequences of, respectively, free and existentially bounded variables occurring in $q(\boldsymbol{y})$ and every atom $\alpha_j(\boldsymbol{d}_j)$ is of the form $A(d)$ or $r(d_1, d_2)$, where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $d, d_1, d_2 \in \mathsf{N_I} \cup \mathsf{N_V}$. Whenever it is not confusing, we sometimes also abbreviate $q(\boldsymbol{y})$ to q. By $\mathsf{term}(q)$ we denote the set of all terms occurring in q and by $\mathsf{obj}(q)$ the set of all free variables. We call q grounded whenever $\mathsf{obj}(q) = \emptyset$. A grounded CQ q is satisfied in $\mathcal{I}$ *iff* there exists a mapping $\mu : \mathsf{term}(q) \mapsto \Delta^{\mathcal{I}}$, with $\mu(d) = d^{\mathcal{I}}$ for every $d \in \mathsf{N_I}$, such that $\mu(d) \in A^{\mathcal{I}}$ and $(\mu(d_1), \mu(d_2)) \in r^{\mathcal{I}}$ for every $A(d)$ and $r(d_1, d_2)$ in q. Further, we say that q is *entailed* by $\mathcal{T}, \mathcal{A}$, denoted as $\mathcal{T}, \mathcal{A} \models q$ *iff* q is satisfied in every model of $\mathcal{T}, \mathcal{A}$. Whenever $\emptyset, \mathcal{A} \models q$ we also write $\mathcal{A} \models q$. An *answer* to q is a mapping $\sigma : \mathsf{obj}(q) \mapsto \mathsf{N_I}$. By $\sigma(q)$ we denote the result of uniformly substituting every occurrence of x in q with $\sigma(x)$, for every $x \in \mathsf{obj}(q)$. An answer σ is called *certain* over $\mathcal{T}$, $\mathcal{A}$ *iff* $\sigma(q)$ is entailed by $\mathcal{T}, \mathcal{A}$.

A prominent property of CQs is their *first-order* (FO) *rewritability* in OWL 2 QL, formally defined as follows.

[2] See `http://www.w3.org/TR/owl2-profiles/#OWL_2_QL` for full details.

Definition 1 (FO Rewritability [8]). *For every CQ q and a TBox $\mathcal{T}$, there exists a FO formula $q^{\mathcal{T}}$, called the* FO rewriting *of q in $\mathcal{T}$, such that for every ABox $\mathcal{A}$ and answer σ to q, it holds that:*

$$\mathcal{T}, \mathcal{A} \models \sigma(q) \ \text{ iff } \ db(\mathcal{A}) \Vdash \sigma(q^{\mathcal{T}}),$$

where $\Vdash$ is the FO satisfaction relation and $db(\mathcal{A})$ denotes $\mathcal{A}$ considered as a database/FO interpretation, i.e., a structure $(\mathsf{N_I}, \cdot^{\mathcal{D}})$, where $\mathsf{N_I}$ is the data domain and $\cdot^{\mathcal{D}}$ is an interpretation function defined as $\alpha^{\mathcal{D}} = \{\boldsymbol{a} \mid \alpha(\boldsymbol{a}) \in \mathcal{A}\}$, for every $\alpha \in \mathsf{N_C} \cup \mathsf{N_R}$.

By the standard techniques the FO rewriting of q in $\mathcal{T}$ is a union of conjunctive queries, i.e., a formula $q^{\mathcal{T}}(\boldsymbol{y}) = \bigvee_{1 \leq i \leq m} q_i(\boldsymbol{y})$, where every $q_i(\boldsymbol{y})$ is a CQ. FO rewritability is particularly significant from the practical perspective. It implies that answering CQs in OWL 2 QL can be effectively performed in existing RDBMSs via a translation to SQL, as the ontological component in the task can be always compiled out in the query rewritting, without loss of soundness or completeness. As a theoretical corollary, it follows also that the data complexity of query answering in this setup is AC^0, as in first-order logic (FOL).

In this work, we focus exclusively on OWL 2 QL TBoxes, even though some of the presented results should clearly transfer to other fragments of DLs warranting the FO rewritability property for CQs. Without always stating it explicitly, we assume that every TBox or ontology mentioned in the remainder of this paper is expressed in OWL 2 QL.

2.2 Ontology-Based Access to Temporal Data

In this paper, we study ontology-based access to temporal data, in the sense of an extension to the OBDA paradigm whose prototypical application could be illustrated with the following scenario.

Consider a temporal database (TDB) presented in Table 1, with columns `from` and `to` marking the limits of the validity periods of the respective records. Such databases can be naturally mapped to (virtual) temporal ABoxes, such as given in Table 2. Our goal is to define a dedicated language for querying temporal ABoxes, which would combine support for two essential functionalities: representation of temporal constraints over the validity periods of data and semantically enhanced access to that data via ontologies. For instance, given the ontology $\mathcal{T} = \{Emp \sqsubseteq Person, department \sqsubseteq worksAt, location \sqsubseteq basedIn\}$, the language should be able to support queries such as:

> (**Q**) *Find all persons X and times Y, such that X worked in a department based in Barcelona during Y and in a department based in Madrid some time earlier.*

The expected set of answers should then include *e1* as X with the associated period $[1999, 2000]$ as Y. The practical rationale behind ontology-based access to temporal data defined in this way is to eventually enable such queries to be translated to SQL and answered within existing RDBMSs.

Table 1. SQL:2011 temporal database

Emp				
`id`	`name`	`department`	`from`	`to`
e1	john	d1	1998	2000
e1	john	d3	2000	2003
e2	mark	d2	1999	2002

Dep				
`id`	`type`	`location`	`from`	`to`
d1	financial	madrid	1998	1999
d1	financial	barcelona	1999	2003
d2	hr	barcelona	2000	2003
d3	hq	london	2000	2003

Table 2. Temporal ABoxes corresponding to the temporal relations in Table 1

$[1998, 2000] : \mathit{Emp}(\mathit{e1})$	$[1998, 1999] : \mathit{Dep}(\mathit{d1})$
$[1998, 2000] : \mathit{name}(\mathit{e1}, \mathit{john})$	$[1998, 1999] : \mathit{type}(\mathit{d1}, \mathit{financial})$
$[1998, 2000] : \mathit{department}(\mathit{e1}, \mathit{d1})$	$[1998, 1999] : \mathit{location}(\mathit{d1}, \mathit{madrid})$
...	...

Formally, the temporal data model under consideration is grounded in the point- and (derived) interval-based time domains.

Definition 2 (Time Domain, Time Intervals). *A* time domain *is a tuple $\mathfrak{T} = (T, <)$, where T is a nonempty set of elements called* time points *and $<$ is an irreflexive, linear ordering on T. A* time interval $\tau = [\tau^-, \tau^+]$ *over $\mathfrak{T}$ is a set of time points $\{t \in T \mid \tau^- \leq t \leq \tau^+\}$, where $\tau^-, \tau^+ \in T$, such that $\tau^- \leq \tau^+$. The points τ^- and τ^+ are called the* beginning *and the* end *of τ.*[3] *The set of all time intervals over $\mathfrak{T}$ is denoted by I.*

In the SQL:2011 standard, every temporal relation extends a non-temporal one with two additional attributes storing the beginning and the end time of the validity period of a given tuple [20] — exactly as in the example from Table 1. Such a model supports a representation of so-called concrete TDBs, i.e., finite syntactic encodings of temporal data. The actual meaning of these encodings is captured by possibly infinite abstract TDBs [12]. In the OBDA setting these two notions translate naturally into the corresponding types of ABoxes.

Definition 3 (Concrete and Abstract Temporal ABoxes). *A* temporal assertion *is an expression $\tau : \alpha(\boldsymbol{a})$, where $\alpha(\boldsymbol{a})$ is an ABox assertion and $\tau \in I$, stating that $\alpha(\boldsymbol{a})$ is valid in every time point in τ. A* concrete temporal ABox *(CTA) $\mathfrak{A}$ is a finite set of temporal assertions. For a concrete temporal ABox $\mathfrak{A}$ there exists a corresponding* abstract temporal ABox *(ATA), obtained by means of a mapping $\|\cdot\|$, such that $\|\mathfrak{A}\| = (\mathfrak{A}_t)_{t \in T}$, where $\mathfrak{A}_t = \{\alpha(\boldsymbol{a}) \mid \tau : \alpha(\boldsymbol{a}) \in \mathfrak{A}$ and $t \in \tau\}$.*

[3] Note that SQL:2011 adopts a closed-open semantics for the validity periods, i.e., for any $\tau \in I$ it holds that $\tau^- \in \tau$ and $\tau^+ \notin \tau$. This a technically insignificant difference which we omit here for clarity of presentation.

The link between concrete temporal ABoxes and SQL:2011 TDBs is defined in a strict analogy to the non-temporal OBDA.

Definition 4 (TDB). *Let $\mathfrak{A}$ be a CTA over the time domain $\mathfrak{T} = (T, <)$. By $tdb(\mathfrak{A})$ we denote $\mathfrak{A}$ considered as a* temporal database *(TDB) over the signature $\Gamma = \{R_\alpha \mid \alpha \in \mathsf{N_C} \cup \mathsf{N_R}\}$, i.e., the structure $(\mathsf{N_I}, T, <, \cdot^{\mathcal{D}})$, where $\mathsf{N_I}$ is the data domain, $(T, <)$ is the time domain, and $\cdot^{\mathcal{D}}$ is the interpretation defined as $R_\alpha^{\mathcal{D}} = \{(\boldsymbol{a}, \tau^-, \tau^+) \mid \tau : \alpha(\boldsymbol{a}) \in \mathfrak{A}\}$, for every $\alpha \in \mathsf{N_C} \cup \mathsf{N_R}$.*

3 Temporal Query Language

The *Temporal Query Language* (TQL), presented in this section, is defined using a generic construction method for temporal query languages in DLs, explored also in [16,3,7,18]. TQL is a combination of a temporal language with CQs, obtained by substituting CQs for the atoms in temporal formulas. This design allows for very flexible interleaving of data queries with temporal constraints, while benefiting from the expressive power of both components. As the temporal language we use first-order monadic logic of orders, which is at least as expressive as most common linear temporal logics [23]. Two specific characteristics of TQL, which distinguish it from other similar proposals, are:

– the use of an interval-based variant of the temporal language, rather than a point-based, which enables direct querying of concrete TDBs, without requiring intermediate translations, such as studied in [25];
– the use of the epistemic semantics for embedding CQs in the temporal language, as suggested in [16], which renders the language more expressive and computationally well-behaved, as explained further.

By $\mathsf{I_V}$ we denote a countably infinite set of variables ranging over I.

Definition 5 (TQL). Temporal query language *(TQL) is the smallest set of formulas induced by the grammar:*

$$\psi \ ::= [q](u) \ \mid \ u^* < v^* \ \mid \ \neg\psi \ \mid \ \psi_1 \wedge \psi_2 \ \mid \ \exists y.\psi$$

where q is a CQ, $u, v \in I \cup \mathsf{I_V}$, $y \in \mathsf{I_V}$, and $* \in \{-, +\}$. An i-substitution is a mapping $\pi : I \cup \mathsf{I_V} \mapsto I$ such that $\pi(\tau) = \tau$, for every $\tau \in I$. By $\mathsf{obj}(\psi)$ we denote the set of free individual variables and by $\mathsf{int}(\psi)$ the set of free interval variables in ψ. A TQL formula ψ is grounded *iff $\mathsf{obj}(\psi) = \mathsf{int}(\psi) = \emptyset$. The satisfaction relation for TQL formulas, w.r.t. a TBox $\mathcal{T}$, a CTA $\mathfrak{A}$, and an i-substitution π, is defined inductively as follows:*

$$\begin{array}{llll}
(\dagger) & \mathcal{T}, \mathfrak{A}, \pi \models [q](u) & \text{iff} & \mathcal{T}, \mathfrak{A}_t \models q, \textit{ for every } t \in \pi(u), \\
& \mathcal{T}, \mathfrak{A}, \pi \models u^* < v^* & \text{iff} & \pi(u)^* < \pi(v)^*, \\
& \mathcal{T}, \mathfrak{A}, \pi \models \neg\psi & \text{iff} & \mathcal{T}, \mathfrak{A}, \pi \not\models \psi, \\
& \mathcal{T}, \mathfrak{A}, \pi \models \psi_1 \wedge \psi_2 & \text{iff} & \mathcal{T}, \mathfrak{A}, \pi \models \psi_1 \textit{ and } \mathcal{T}, \mathfrak{A}, \pi \models \psi_2, \\
& \mathcal{T}, \mathfrak{A}, \pi \models \exists y.\psi & \text{iff} & \textit{there exists } \tau \in I, \textit{ such that} \\
& & & \mathcal{T}, \mathfrak{A}, \pi[y \mapsto \tau] \models \psi,
\end{array}$$

where $\pi[y \mapsto \tau]$ *denotes the i-substitution exactly as* π *except for that we fix* $\pi(y) = \tau$*. We say that* $\mathcal{T}, \mathfrak{A}$ entail *a grounded TQL formula* ψ*, denoted as* $\mathcal{T}, \mathfrak{A} \models \psi$*,* iff *there exists an i-substitution* π*, such that* $\mathcal{T}, \mathfrak{A}, \pi \models \psi$*.*

TQL formulas with free variables can naturally serve as queries over concrete temporal ABoxes. We refer to such formulas as *concrete TQL queries* (CTQs). As an example of a CTQ, consider a rephrasing of the query (**Q**) from Section 2.2:

$$\psi(x, y) := [\exists z.(Person(x) \land worksAt(x, z) \land basedIn(z, barcelona))](y) \land \\ \exists v.(v^+ < y^- \land [\exists z.(worksAt(x, z) \land basedIn(z, madrid))](v))$$

As one of its answers, $\psi(x, y)$ should return $\{x \mapsto e1, y \mapsto [1999, 2000]\}$. The certain answer semantics for such queries is defined as expected.

Definition 6 (CTQ Answering). *Let* $\mathcal{T}$ *be a TBox,* $\mathfrak{A}$ *a CTA and* ψ *a CTQ with free variables* $\mathsf{obj}(\psi)$ *and* $\mathsf{int}(\psi)$*. An* answer *to* ψ *is a mapping* σ *such that* $\sigma : \mathsf{obj}(\psi) \mapsto \mathsf{N_I}$ *and* $\sigma : \mathsf{int}(\psi) \mapsto I$*. By* $\sigma(\psi)$ *we denote the result of uniformly substituting every occurrence of* x *in* ψ *with* $\sigma(x)$*, for every* $x \in \mathsf{obj}(\psi) \cup \mathsf{int}(\psi)$*. An answer* σ *is called* certain *over* $\mathcal{T}$*,* $\mathfrak{A}$ iff $\mathcal{T}, \mathfrak{A} \models \sigma(\psi)$*.*

The key to the design of TQL is condition (†), in Definition 5, which ensures the epistemic interpretation of the CQs embedded in TQL queries. The formula $[q](\tau)$, for a grounded CQ q and $\tau \in I$, reads as "*q is entailed in all time points in τ*". Analogically, $\neg[q](\tau)$ is interpreted as negation-as-failure: "*it is not true that q is entailed in all time points in τ*". This approach of combining FO-based query languages has been originally proposed by Calvanese et al. [9], giving rise to a family of lightweight query languages, which permit well-behaved, modular answering algorithms and support the use of negation (interpreted as negation-as-failure), which otherwise easily leads to undecidability in the context of querying DL ontologies.

As a consequence, condition (†) can be effectively replaced with its equivalent, which involves the standard FO rewritability techniques in the sense of Definition 1:

$$\mathcal{T}, \mathfrak{A}, \pi \models [q](u) \quad \text{iff} \quad db(\mathfrak{A}_t) \Vdash q^{\mathcal{T}}, \text{ for every } t \in \pi(u),$$

where $q^{\mathcal{T}}$ is an FO rewriting of q in $\mathcal{T}$. What it eventually means, is that all occurrences of CQs in a CTQ can be replaced with their FO rewritings, so that the ontology $\mathcal{T}$ can be dropped while the formula can be evaluated exclusively over the temporal ABox seen as a sequence of FO interpretations. However, such point-based rewriting strategy, also explored in [7], is highly impractical when applied over concrete TDBs, as it necessitates either unfolding a concrete TDB into an abstract one, or a further translation from point-based to an interval-based language. Instead, here we pursue a direct interval-based approach, which requires a more sophisticated rewriting technique, capable of handling the well-known problems of computing temporal joins and coalescing, as explained in detail in the following section.

4 Query Rewriting

We start by defining a two-sorted first-order language, tailored specifically for talking about temporal relations of TDBs.

Definition 7 (2FO). *The language of* two-sorted first-order logic *(2FO) over the signature $\Gamma = \{R_1, R_2, \ldots\}$ is the smallest set of formulas induced by the grammar:*

$$\varphi ::= R(d_1, \ldots, d_n, t_1, t_2) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid t_1 < t_2 \mid \exists x.\varphi \mid \exists y.\varphi$$

where $R \in \Gamma$, $d_1, \ldots, d_n \in \mathsf{N_I} \cup \mathsf{N_V}$, $t_1, t_2 \in T \cup \mathsf{T_V}$, $x \in \mathsf{N_V}$ and $y \in \mathsf{T_V}$. A d-substitution δ is a mapping $\delta : \mathsf{N_I} \cup \mathsf{N_V} \mapsto \mathsf{N_I}$, such that $\delta(a) = a$ for every $a \in \mathsf{N_I}$. A t-substitution ν is a mapping $\nu : T \cup \mathsf{T_V} \mapsto T$, such that $\nu(t) = t$ for every $t \in T$. A 2FO formula φ is called grounded *whenever it does not have any free variables. The satisfaction relation for 2FO formulas, w.r.t. a TDB $tdb(\mathfrak{A}) = (\mathsf{N_I}, T, <, \cdot^{\mathcal{D}})$, a d-substitution δ and a t-substitution ν, is defined inductively as follows:*

$tdb(\mathfrak{A}), \delta, \nu \Vdash R(d_1, \ldots, d_n, t_1, t_2)$	iff	$(\delta(d_1), \ldots, \delta(d_n), \nu(t_1), \nu(t_2)) \in R^{\mathcal{D}}$,
$tdb(\mathfrak{A}), \delta, \nu \Vdash \neg\varphi$	iff	$tdb(\mathfrak{A}), \delta, \nu \nVdash \varphi$,
$tdb(\mathfrak{A}), \delta, \nu \Vdash \varphi_1 \wedge \varphi_2$	iff	$tdb(\mathfrak{A}), \delta, \nu \Vdash \varphi_1$ *and* $tdb(\mathfrak{A}), \delta, \nu \Vdash \varphi_2$,
$tdb(\mathfrak{A}), \delta, \nu \Vdash t_1 < t_2$	iff	$\nu(t_1) < \nu(t_2)$,
$tdb(\mathfrak{A}), \delta, \nu \Vdash \exists x.\varphi$	iff	*for $x \in \mathsf{N_V}$, there exists $a \in \mathsf{N_I}$ such that $tdb(\mathfrak{A}), \delta[x \mapsto a], \nu \Vdash \varphi$,*
$tdb(\mathfrak{A}), \delta, \nu \Vdash \exists y.\varphi$	iff	*for $y \in \mathsf{T_V}$, there exists $t \in T$ such that $tdb(\mathfrak{A}), \delta, \nu[y \mapsto t] \Vdash \varphi$,*

where $\delta[x \mapsto a]$ ($\nu[y \mapsto t]$) denotes the substitution exactly as δ (ν) except for that we fix $\delta(x) = a$ ($\nu(y) = t$). We say that a 2FO formula φ is satisfied *in $tdb(\mathfrak{A})$, denoted as $tdb(\mathfrak{A}) \Vdash \varphi$,* iff *there exist d-/t-substitutions δ, ν, such that $tdb(\mathfrak{A}), \delta, \nu \Vdash \varphi$.*

Next, we define the rules for rewriting CTQs into 2FO formulas.

Definition 8 (2FO Rewriting of CTQs). *The* 2FO rewriting *of a CTQ ψ is a formula $\lceil\psi\rceil^{2FO}$ obtained from ψ by applying the transformation $\lceil\cdot\rceil^{2FO}$, defined inductively as follows:*

$$\begin{aligned}
\lceil [q(\boldsymbol{d})](u) \rceil^{2FO} &= \exists t_1, t_2.(R^{coal}_{q^{\mathcal{T}}}(\boldsymbol{d}, t_1, t_2) \wedge t_1 \leq u^- \wedge u^+ \leq t_2),\\
\lceil u^* < v^* \rceil^{2FO} &= u^* < v^*,\\
\lceil \neg\psi \rceil^{2FO} &= \neg\lceil\psi\rceil^{2FO},\\
\lceil \psi_1 \wedge \psi_2 \rceil^{2FO} &= \lceil\psi_1\rceil^{2FO} \wedge \lceil\psi_2\rceil^{2FO},\\
\lceil \exists y.\psi \rceil^{2FO} &= \exists y^-, y^+.(y^- \leq y^+ \wedge \lceil\psi\rceil^{2FO}),
\end{aligned}$$

where the involved syntactic abbreviations are as follows:

$$\begin{aligned}
R^{coal}_{q^{\mathcal{T}}}(\boldsymbol{d}, u, v) \triangleq\ & \exists t_1, t_2.R_{q^{\mathcal{T}}}(\boldsymbol{d}, u, t_1) \wedge R_{q^{\mathcal{T}}}(\boldsymbol{d}, t_2, v)\ \wedge\\
& \neg\exists t_3, t_4.(R_{q^{\mathcal{T}}}(\boldsymbol{d}, t_3, t_4) \wedge t_3 < u \wedge u \leq t_4)\ \wedge\\
& \neg\exists t_3, t_4.(R_{q^{\mathcal{T}}}(\boldsymbol{d}, t_3, t_4) \wedge t_3 \leq v \wedge v < t_4)\ \wedge\\
& \neg\exists t_3, t_4.(R_{q^{\mathcal{T}}}(\boldsymbol{d}, t_3, t_4) \wedge u < t_3 \wedge t_4 \leq v\ \wedge\\
& \neg\exists t_5, t_6.(R_{q^{\mathcal{T}}}(\boldsymbol{d}, t_5, t_6) \wedge t_5 < t_3 \wedge t_3 \leq t_6))
\end{aligned} \tag{1}$$

where for $q^{\mathcal{T}} = \bigvee_{1\leq i\leq m} q_i$*:*

$$R_{q^{\mathcal{T}}}(\boldsymbol{d}, u, v) \triangleq \bigvee_{1\leq i\leq m} R_{q_i}(\boldsymbol{d}, u, v) \tag{2}$$

and for every $q_i(\boldsymbol{d}) = \exists \boldsymbol{x}.(\bigwedge_{1\leq j\leq n} \alpha_j(\boldsymbol{d}_j))$*:*

$$\begin{aligned} R_{q_i}(\boldsymbol{d}, u, v) \triangleq \exists t_1, \ldots, t_{2n}.\exists \boldsymbol{x}. \bigwedge_{1\leq j\leq n} (R_{\alpha_j}(\boldsymbol{d}_j, t_j, t_{n+j})) \wedge \\ u = \max(t_1, \ldots, t_n) \wedge v = \min(t_{n+1}, \ldots, t_{2n}) \wedge u \leq v \end{aligned} \tag{3}$$

where:

$$\begin{aligned} u = \max(t_1, \ldots, t_n) &\triangleq \bigwedge_{1\leq i\leq n} ((\bigwedge_{1\leq j\leq n} t_j \leq t_i) \rightarrow u = t_i) \\ v = \min(t_1, \ldots, t_n) &\triangleq \bigwedge_{1\leq i\leq n} ((\bigwedge_{1\leq j\leq n} t_i \leq t_j) \rightarrow v = t_i) \end{aligned} \tag{4}$$

The pivotal part of CTQ rewriting is the translation of the embedded CQs. In this respect, the proposed approach builds directly on the standard FO rewritings of CQs, which are obtainable via existing techniques [8]. Given an FO rewriting $q^{\mathcal{T}}$ of a CQ q, all atoms in $q^{\mathcal{T}}$ are temporalized (3) and further incorporated in special formula templates (1)-(4), in order to meet two key challenges inherent to querying concrete TDBs:

- computing *temporal joins*, i.e., identifying maximal time intervals over which conjunctions of atoms are satisfied,
- applying *coalescing*, i.e., merging overlapping and adjacent intervals for the (intermediate) query results.

To illustrate these issues and the roles played by the formula templates, consider the example presented in Figure 1, which addresses the following setup:

CTA:	$\mathfrak{A} = \{[1, 7] : B(a), [1, 3] : C(a), [3, 10] : C(a), [6, 12] : D(a)\}$,
TBox:	$\mathcal{T} = \{D \sqsubseteq B\}$,
CTQ:	$[q(a)](u)$, where $q(x) = B(x) \wedge C(x)$ and $u = [1, 10]$.

Under the assumed TBox, the FO rewriting of q can be formulated as $q^{\mathcal{T}}(x) = q_1(x) \vee q_2(x)$, where $q_1(x) = B(x) \wedge C(x)$ and $q_2(x) = D(x) \wedge C(x)$. By Definition 8, the satisfaction of condition $tdb(\mathfrak{A}) \Vdash [q(a)](u)$ is equivalent to verifying whether $(a, t_1, t_2) \in (R^{coal}_{q^{\mathcal{T}}})^{\mathcal{D}}$, for some $t_1, t_2 \in T$, such that $t_1 \leq u^- \leq u^+ \leq t_2$. Computing $(R^{coal}_{q^{\mathcal{T}}})^{\mathcal{D}}$ can be conceptually divided into three consecutive phases.

Firstly, we compute temporal joins over the sets of ABox assertions entailing each CQ q_i, i.e., we identify all intervals τ, such that there exist a set of assertions $S = \{\tau_j : \alpha_j(\boldsymbol{d}_j) \in \mathfrak{A}\}$, where the atoms $\alpha_j(\boldsymbol{d}_j)$ in S provide the exact matches for the conjuncts of q_i, while $\bigcap_j \tau_j = \tau$. Relation R_{q_i}, defined via formula template (3), augmented with (4), which fixes an FO formalization of the functions max and min, selects exactly such intervals for each answer to each q_i. In our example, R_{q_1} contains tuples $(a, 1, 3)$ and $(a, 3, 7)$, while R_{q_2}

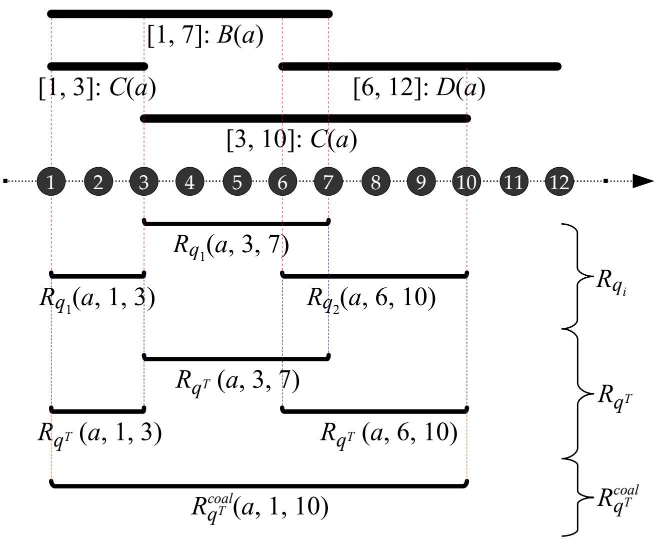

Fig. 1. Computing $R_{q_i}, R_{q^\mathcal{T}}$ and $R^{coal}_{q^\mathcal{T}}$, for $q(x) = B(x) \wedge C(x)$ and $\mathcal{T} = \{D \sqsubseteq B\}$

tuple $(a, 6, 10)$. In the second phase, captured by the template (2), all such tuples become automatically instances of the common relation $R_{q^\mathcal{T}}$. Finally, $R_{q^\mathcal{T}}$ is coalesced, resulting in the relation $R^{coal}_{q^\mathcal{T}}$. This phase is executed by means of the template (1), which is based on a straightforward FO formalization of the coalescing mechanism, known in the context of TDBs [6]. Technically, for each tuple $\boldsymbol{a}$ comprising an answer to q, the relation $R^{coal}_{q^\mathcal{T}}$ selects the minimal and maximal time points, such that for every point t between the two there exists a tuple $(\boldsymbol{a}, t_1, t_2) \in (R_{q^\mathcal{T}})^\mathcal{D}$ with $t_1 \leq t \leq t_2$. In the presented example, $R^{coal}_{q^\mathcal{T}}$ contains exactly one tuple $(a, 1, 10)$, which is the result of coalescing tuples $(a, 1, 3), (a, 3, 7)$ and $(a, 6, 10)$ in $R_{q^\mathcal{T}}$. Consequently, the CTQ $[q(a)](u)$ evaluates to true over $\mathcal{T}$ and $\mathfrak{A}$.

Since 2FO queries, in the shape defined above, can be naturally considered as formulas of the relational calculus, the final step of restating them into the SQL syntax can be carried out following broadly adopted translation strategies [17]. In practice, to ensure appropriate quantification over the time domain in RDBMSs, one would usually need to include it as an explicitly represented (and stored in the TDB) monadic relation R_T, such that $R_T^\mathcal{D} = T$, which should be further introduced in the translation of the temporal quantifiers from the Definition 1, as follows:

$$\lceil \exists y.\psi \rceil^{2FO} \quad = \quad \exists y^-, y^+.(R_T(y^-) \wedge R_T(y^+) \wedge y^- \leq y^+ \wedge \lceil \psi \rceil^{2FO}).$$

Accordingly, every free temporal variable $x \in \mathsf{T_V}$ occurring in a 2FO query should be additionally guarded by the restriction $R_T(x)$ in the resulting translation, in order to be properly handled by the substitution mechanisms implemented in typical RDBMSs.

Agreeably, even though the presented rewriting is mathematically correct, as we demonstrate in the following part, it can be expected to be suboptimal in terms of the query answering efficiency. For instance, the naive coalescing mechanism, captured by the formula template (1), is known to be highly inefficient and can be substantially improved using more sophisticated approaches [26]. The usefulness of this and other potential optimizations, on which we also shortly remark at the end of the next section, can be ultimately assessed only on the grounds of empirical evaluation, which is outside the scope of the current work.

5 Query Answering via 2FO Rewriting

The next theorem ensures correctness of CTQ answering via the 2FO rewriting.

Theorem 1 (Correctness of 2FO Rewriting). *For every TBox $\mathcal{T}$, CTA $\mathfrak{A}$, CTQ ψ, and an answer σ to ψ, it holds that:*

$$\mathcal{T}, \mathfrak{A} \models \sigma(\psi) \;\text{ iff }\; tdb(\mathfrak{A}) \Vdash \lceil \sigma(\psi) \rceil^{2FO}.$$

The proof, presented in the appendix, follows by structural induction over the syntactic cases addressed in the rewriting rules. In the technically most demanding case of $\lceil [q(\boldsymbol{d})](u) \rceil^{2FO}$ we essentially formalize the discussion from the previous section. The remaining ones are largely straightforward.

Importantly, the definitions of CTQs and their 2FO rewritings, guarantee that query answering remains insensitive to the particular ways the CTAs might encode temporal data, as long as the data is semantically equivalent. This result is due to the fact that the semantics of CTQs is defined in terms of the underlying ATAs, hence whenever $\|\mathfrak{A}\| = \|\mathfrak{A}'\|$, for any two CTAs $\mathfrak{A}$ and $\mathfrak{A}'$, the answers over them must always coincide.

Theorem 2 (CTQs are $\|\cdot\|$-Generic). *For every two CTAs $\mathfrak{A}$ and $\mathfrak{A}'$ over a time domain $\mathfrak{T} = (T, <)$ such that $\|\mathfrak{A}\| = \|\mathfrak{A}'\|$, a TBox $\mathcal{T}$, a CTQ ψ and an answer σ to ψ, it holds that:*

1. $\mathcal{T}, \mathfrak{A} \models \sigma(\psi)$ iff $\mathcal{T}, \mathfrak{A}' \models \sigma(\psi)$,
2. $tdb(\mathfrak{A}) \Vdash \lceil \sigma(\psi) \rceil^{2FO}$ iff $tdb(\mathfrak{A}') \Vdash \lceil \sigma(\psi) \rceil^{2FO}$.

Further, we address the complexity and algorithmic aspects of the CTQ entailment. We start by observing that, similarly as in the case of CQ rewriting, 2FO rewritings of CTQs can be in principle exponential in the size of the original queries. In fact, as the next proposition shows, this exponential blow-up is exclusively due to the embedded CQs, as the extra temporal layer itself adds only linearly to the overall size of the resulting 2FO formulas. This observation follows directly by scrutinizing the rewriting rules from Definition 8.

Proposition 1 (Size of Rewriting). *Let $\mathcal{T}$ be a TBox, and ψ a CTQ with $q_1, \ldots, q_n$ being all the CQs occurring in ψ. Then the size of $\lceil \psi \rceil^{2FO}$ is linear in the joint size of ψ and the FO rewritings $q_1^{\mathcal{T}}, \ldots, q_n^{\mathcal{T}}$ of the CQs.*

Algorithm 1. Incremental computation of relations $R^{coal}_{q\mathcal{T}}$

```
Input: CTA $\mathfrak{A}$, CTQ $\psi$
Output: A database $\mathcal{R} = (\mathsf{N_I}, T, <, \cdot^{\mathcal{D}})$ over the signature $\{R^{coal}_{q\mathcal{T}} \mid q \in \mathsf{cq}(\psi)\}$
 1: for all $q \in \mathsf{cq}(\psi)$ do
 2:    for all $q_i(\boldsymbol{a}) \in \mathsf{cq}(q^{\mathcal{T}})$ do
 3:       for all $t_1, t_2 \in T$ such that $tdb(\mathfrak{A}) \Vdash R_{q_i}(\boldsymbol{a}, t_1, t_2)$ do
 4:          $(R^{coal}_{q\mathcal{T}})^{\mathcal{D}} := (R^{coal}_{q\mathcal{T}})^{\mathcal{D}} \cup \{(\boldsymbol{a}, t_1, t_2)\}$
 5:          while applicable do {coalescing}
 6:             for all $(\boldsymbol{a}, t_1, t_2), (\boldsymbol{a}, t_3, t_4) \in (R^{coal}_{q\mathcal{T}})^{\mathcal{D}}$ do
 7:                if $[t_1, t_2] \cap [t_3, t_4] \neq \emptyset$ then
 8:                   $(R^{coal}_{q\mathcal{T}})^{\mathcal{D}} := (R^{coal}_{q\mathcal{T}})^{\mathcal{D}} \setminus \{(\boldsymbol{a}, t_1, t_2), (\boldsymbol{a}, t_3, t_4)\}$
 9:                   $(R^{coal}_{q\mathcal{T}})^{\mathcal{D}} := (R^{coal}_{q\mathcal{T}})^{\mathcal{D}} \cup \{(\boldsymbol{a}, \min(t_1, t_3), \max(t_2, t_4))\}$
10:                end if
11:             end for
12:          end while
13:       end for
14:    end for
15: end for
```

Clearly, the fragment of two-sorted first-order logic used for the CTQ rewriting is as expressive as FOL. At the same time, it is obviously not more expressive than that, as one can apply a commonly known, linear reduction, involving the introduction of two designated predicates for representing the respective domains, which are used to guard the scopes of the two sorts of quantifiers (*cf.* the last paragraph of the previous section). Based on these observations, we obtain two results concerning the complexity of CTQ query entailment.

Corollary 1 (Data Complexity). *The data complexity of CTQ entailment in CTAs in the presence of TBoxes, over finite time domains, is in* AC^0.

The restriction to finite time domains in this and the following case is a natural strategy of ensuring that the first-order structures, over which queries are evaluated, are indeed finite and, consequently, that model-checking can be effectively performed over them. Note, that in terms of data complexity, deciding CTQ entailment via 2FO rewriting remains precisely as hard as deciding CQ entailment via FO rewriting. The combined complexity, however, increases from NP- to PSpace-complete.

Theorem 3 (Combined Complexity). *The combined complexity of CTQ entailment over CTAs in the presence of TBoxes, over finite time domains, is* PSpace*-complete.*

The result transfers from the entailment problem for boolean FO queries over relational databases, which is known to be PSpace-complete [10]. However, due to the exponential blow-up in the size of the 2FO rewritings, explained in Proposition 1, answering CTQs in PSpace requires a somewhat more sophisticated strategy than just a straightforward evaluation of the rewritten queries over a

TDB. In the approach described in the following proof, we decide the entailment of a grounded CTQ ψ, by first incrementally computing the interpretations of relations $R^{coal}_{q^{\mathcal{T}}}$, for each CQ q embedded in ψ, and then evaluate over them a restricted 2FO rewriting of ψ of at most polynomial size.

Proof. Let $\mathsf{cq}(\psi)$ denote all CQs occurring in ψ and grounded as in ψ, and let $\mathsf{cq}(q^{\mathcal{T}})$, for every $q \in \mathsf{cq}(\psi)$, be the set of all CQs comprising the FO rewriting $q^{\mathcal{T}}$. With every such $q_i(\boldsymbol{a}) \in \mathsf{cq}(q^{\mathcal{T}})$, we associate the (query) formula $R_{q_i}(\boldsymbol{a}, y_1, y_2)$ defined via the templates (3) and (4). Note that every such formula is linear in the size of ψ. Algorithm 1 constructs a TDB $\mathcal{R} = (\mathsf{N_I}, T, <, \cdot^{\mathcal{D}})$ over the signature $\{R^{coal}_{q^{\mathcal{T}}} \mid q \in \mathsf{cq}(\psi)\}$, by collecting and coalescing all matches to the formulas $R_{q_i}(\boldsymbol{a}, y_1, y_2)$ over $tdb(\mathfrak{A})$. Observe that by applying coalescing on $R^{coal}_{q^{\mathcal{T}}}$ every time another tuple is added, we guarantee that the size of each $(R^{coal}_{q^{\mathcal{T}}})^{\mathcal{D}}$, and therefore of $\mathcal{R}$, is linear in the size of T, and in fact cannot be larger then $\mathfrak{A}$. Therefore the algorithm runs in PSpace. Once $\mathcal{R}$ is computed, we decide $\mathcal{R} \Vdash \lceil \psi \rceil^{2FO}_{R}$, where $\lceil \cdot \rceil^{2FO}_{R}$ is a rewriting specified exactly as $\lceil \cdot \rceil^{2FO}$ in Definition 8, but without employing any of the formula templates (1)-(4), and instead, retaining all $R^{coal}_{q^{\mathcal{T}}}$ as actual predicates in the resulting 2FO formulas. Clearly, $\lceil \psi \rceil^{2FO}_{R}$ is linear in the size of ψ. Since deciding $\mathcal{R} \Vdash \lceil \psi \rceil^{2FO}_{R}$ is a variant of the (PSpace-hard [10]) entailment problem for boolean FO queries over relational databases (observe, that $\lceil \cdot \rceil^{2FO}_{R}$ permits n-ary predicates, for $n \geq 1$), it follows that deciding the CTQ entailment over CTAs in the presence of TBoxes, over finite time domains, is PSpace-complete. Note that since the interpretations of $R^{coal}_{q^{\mathcal{T}}}$ in both $\mathcal{R}$ and $tdb(\mathfrak{A})$ must coincide, the procedure preserves soundness and completeness of query answering. ❑

The query answering algorithm implied by the above decision procedure paves the way towards efficient practical implementations, based on the well-known database technique of *materialized view maintenance* [21], particularly in the context of large or often changing TDBs. By maintaining the intermediate views containing the answers to the embedded CQs, the approach should facilitate more localized and fine-grained updating of the answers in response to database updates and query refinements (be it on the CQ or temporal level), as well as forms of incremental, anytime query answering.

6 Related Work

This work is naturally related to the research on TDBs, conducted largely in the 1990s [11,12]. Although the advancements in that area provide some theoretical grounding for our proposal, they are obviously agnostic about the ontology-based approach on the problem, which we concentrate on here.

The research on temporal extensions to OBDA has been taken up only very recently by Borgwardt et al. [3,7] and Artale et al. [2]. In [3], the authors study the same prototypical scenario as addressed here, but focus on its more foundational aspects. They consider a more expressive DL $\mathcal{ALC}$ as the background ontology language and adopt a less restrictive definition of temporal queries.

This offers a richer setting, yet without apparent application prospects in the context of existing TDBs. The work in [7] is closer aligned with ours. It also considers DL-*Lite* ontologies and studies a rewriting approach based on the use of standard CQ rewriting techniques. However, the authors define their query language without involving the epistemic interpretation of embedded CQs and consider only its positive fragment. It is not difficult to show that under OWL 2 QL, such design leads to a query language strictly less expressive from the one proposed here. Observe, that due to the condition (†), in Definition 5, the class of negation-free CTQs preserves the semantics regardless of the use of the epistemic interpretation of CQs, and so it coincides with the fragment considered in [7]. Furthermore, the authors focus exclusively on abstract TDBs and do not address the problem of direct querying of concrete representations, which is the main goal of our work. In [2], the authors study an orthogonal approach to temporal OBDA. Instead of adding temporal features on the query level, they propose temporal extensions to OWL 2 QL, in the spirit of temporal DLs [1].

A number of other frameworks have been developed for managing the validity time of Semantic Web data represented natively in RDF(S)/OWL languages [15,14,5,22]. These proposals can be seen as parallel to ours, in that they address variants of the same problem, but focus on the use of dedicated Semantic Web technologies, such as SPARQL and RDF triplestores, instead of involving a semantic view on the data managed within traditional RDBMSs. This rises interesting questions about the formal correspondences between the employed reasoning methods, regardless of their practical implementations. Unfortunately, most of these approaches are strongly technology-driven and often fall short of indicating their links to the logic foundations of temporal querying.

7 Conclusions

In this work, we have proposed a principled, yet practical approach to lifting the popular paradigm of OBDA to temporal case. The presented language TQL allows for querying temporal data stored in SQL:2011-compliant databases via a semantic layer of OWL 2 QL ontologies. We believe that this proposal strikes a good balance between the theoretical strength of its formal foundations and the feasibility of practical applications, warranted by the possibility of answering TQL queries in commercially supported RDBMSs via a translation to SQL.

Considering the growing interest in temporal extensions of OBDA, it is critical to continue the formal study of temporal features supported by the SQL standards, temporal extensions to logic-based query languages and ontology languages intended for use in practical OBDA scenarios, and finally, the relationships holding between all of them. In this respect, as part of future research, it is necessary to establish stronger links between the approach pursued here and those proposed recently in [3,7,2], with the prospect of gaining a clearer view on the landscape of technical possibilities regarding the temporal OBDA. In particular, the scenario of querying temporal data, as considered here and in [3,7], under temporalized ontologies, such as introduced by Artale et al. [2], deserves further attention and in-depth study.

References

1. Artale, A., Kontchakov, R., Lutz, C., Wolter, F., Zakharyaschev, M.: Temporalising tractable description logics. In: Proceedings of the Fourteenth International Symposium on Temporal Representation and Reasoning (2007)
2. Artale, A., Kontchakov, R., Wolter, F., Zakharyaschev, M.: Temporal description logic for ontology-based data access. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2013) (2013)
3. Baader, F., Borgwardt, S., Lippmann, M.: Temporalizing ontology-based data access. In: Bonacina, M.P. (ed.) CADE 2013. LNCS, vol. 7898, pp. 330–344. Springer, Heidelberg (2013)
4. Baader, F., Calvanese, D., Mcguinness, D.L., Nardi, D., Patel-Schneider, P.F.: The description logic handbook: theory, implementation, and applications. Cambridge University Press (2003)
5. Batsakis, S., Stravoskoufos, K., Petrakis, E.G.M.: Temporal Reasoning for Supporting Temporal Queries in OWL 2.0. In: König, A., Dengel, A., Hinkelmann, K., Kise, K., Howlett, R.J., Jain, L.C. (eds.) KES 2011, Part I. LNCS, vol. 6881, pp. 558–567. Springer, Heidelberg (2011)
6. Böhlen, M.H., Snodgrass, R.T., Soo, M.D.: Coalescing in temporal databases. IEEE Computer 19, 35–42 (1996)
7. Borgwardt, S., Lippmann, M., Thost, V.: Temporal query answering in the description logic DL-Lite. In: Fontaine, P., Ringeissen, C., Schmidt, R.A. (eds.) FroCoS 2013. LNCS, vol. 8152, pp. 165–180. Springer, Heidelberg (2013)
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning 39(3), 385–429 (2007)
9. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Eql-lite: Effective first-order query processing in description logics. In: Proc. of IJCAI 2007 (2007)
10. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: Proc. of the ACM Symposium on Theory of Computing (STOC 1977) (1977)
11. Chomicki, J.: Temporal query languages: A survey. In: Gabbay, D.M., Ohlbach, H.J. (eds.) ICTL 1994. LNCS, vol. 827, pp. 506–534. Springer, Heidelberg (1994)
12. Chomicki, J., Toman, D.: Temporal Databases. In: Handbook of Temporal Reasoning in Artificial Intelligence (Foundations of Artificial Intelligence), pp. 429–468. Elsevier Science Inc. (2005)
13. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic SHIQ. Journal of Artificial Intelligence Research (2008)
14. Grandi, F.: T-SPARQL: a TSQL2-like temporal query language for RDF. In: Proc. of the International Workshop on Querying Graph Structured Data (2010)
15. Gutierrez, C., Hurtado, C.A., Vaisman, A.A.: Introducing time into RDF. IEEE Transactions onn Knowledge and Data Engineering 19(2), 207–218 (2007)
16. Gutiérrez-Basulto, V., Klarman, S.: Towards a unifying approach to representing and querying temporal data in description logics. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 90–105. Springer, Heidelberg (2012)
17. Kawash, J.: Complex quantification in Structured Query Language (SQL): A tutorial using relational calculus. Journal of Computers in Mathematics and Science Teaching 23(2), 169–190 (2004)

18. Klarman, S., Meyer, T.: Prediction and explanation over DL-Lite data streams. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) LPAR-19 2013. LNCS, vol. 8312, pp. 536–551. Springer, Heidelberg (2013)
19. Klarman, S., Meyer, T.: Querying temporal databases via OWL 2 QL. Tech. rep., CAIR, UKZN/CSIR Meraka (2014), `http://klarman.synthasite.com/resources/KlaMeyRR14.pdf`
20. Kulkarni, K., Michels, J.E.: Temporal features in SQL:2011. SIGMOD Rec. 41(3) (2012)
21. Mohania, M., Konomi, S., Kambayashi, Y.: Incremental maintenance of materialized views. In: Tjoa, A.M. (ed.) DEXA 1997. LNCS, vol. 1308, pp. 551–560. Springer, Heidelberg (1997)
22. Motik, B.: Representing and querying validity time in RDF and OWL: A logic-based approach. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 57(5), 1–62 (2012)
23. Reynolds, M.: The complexity of decision problems for linear temporal logics. Journal of Studies in Logic 3(1) (2010)
24. Snodgrass, R.T., Böhlen, M.H., Jensen, C.S., Steiner, A.: Transitioning temporal support in TSQL2 to SQL3. In: Etzion, O., Jajodia, S., Sripada, S. (eds.) Dagstuhl Seminar 1997. LNCS, vol. 1399, pp. 150–194. Springer, Heidelberg (1998)
25. Toman, D.: Point vs. interval-based query languages for temporal databases. In: Proc. of the Symposium on Principles of Database Systems (PODS 1996) (1996)
26. Zhou, X., Wang, F., Zaniolo, C.: Efficient temporal coalescing query support in relational database systems. In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 676–686. Springer, Heidelberg (2006)