Temporal Query Answering in *DL-Lite*[*]

Stefan Borgwardt, Marcel Lippmann, and Veronika Thost

Theoretical Computer Science, TU Dresden, Germany
{stefborg,lippmann,thost}@tcs.inf.tu-dresden.de

1 Introduction

Context-aware applications try to detect specific situations within a changing environment (e.g. a computer system or air traffic observed by radar) in order to react accordingly. To gain information, the environment is observed by sensors (for a computer system, data about its resources is gathered by the operating system), and the results of sensing are stored in a database. A context-aware application then detects specific predefined situations based on this data (e.g. a high system load) and executes some action (e.g. increases the CPU frequency).

In a simple setting, such an application can be realized by using standard database techniques: the sensor information is stored in a database, and the situations to be recognized are specified as database queries [1]. However, we cannot assume that the sensors provide a complete description of the current state of the environment. Thus, the closed world assumption employed by database systems (i.e. facts not present in the database are assumed to be false) is not appropriate since there may be facts of which the truth is not known. For example, a sensor for certain information might not be available for a moment or not even exist.

In addition, though a complete specification of the environment usually does not exist, often some knowledge about its behavior is available. This knowledge can be used to formulate constraints on the interpretation of the predicates used in the queries, to detect more complex situations. In ontology-based data access (OBDA) [8], domain knowledge is encoded in ontologies using a description logic (DL). In this paper, we consider logics of the *DL-Lite* family, which are lightweight DLs with a low complexity for many reasoning problems [8].

In order to recognize situations that evolve over time, we propose to add a temporal logical component to the queries. We use the operators of the temporal logic LTL, which allows to reason about a linear and discrete flow of time [17]. Usual temporal operators include *next* ($\bigcirc\phi$), *eventually* ($\Diamond\phi$), and *always* ($\Box\phi$), asserting that a property ϕ is true at the next point in time, at some point in the future, and at all time points in the future, respectively. We also use the corresponding past operators $\bigcirc^-$, $\Diamond^-$, and $\Box^-$.

Consider, for example, a collection of servers providing several services. An important task is to migrate services between servers to balance the load. To decide when to migrate, we want to detect certain critical situations. We consider a process to be critical if it has an increasing workload, and at the same time the

[*] Partially supported by DFG SFB 912 (HAEC) and GRK 1763 (QuantLA).

server that hosts it is almost overloaded. We want to detect those processes and servers that were in a critical situation at least twice within the past ten time units, expressed by the query $\bigcirc^{-10}(\Diamond(\mathsf{Critical}(x,y) \wedge \bigcirc\Diamond\mathsf{Critical}(x,y)))$, where

$$\mathsf{Critical}(x,y) := \mathsf{Server}(x) \wedge \mathsf{Process}(y) \wedge \mathsf{executes}(x,y) \wedge \mathsf{Running}(y) \wedge \mathsf{IncreasingWorkload}(y) \wedge \mathsf{AlmostOverloaded}(x).$$

In this example, it is essential that future and past operators can be nested arbitrarily. One might argue that, as we are looking at the time line from the point of view of the current time point, and nothing is known about the future, having only past operators is sufficient. We will even show that in our setting it is always possible to construct an equivalent query using only past operators. However, the resulting query is not very concise and it is not easy to see the situation that is to be recognized. This is similar to propositional LTL, where eliminating the past operators from a formula incurs a blowup that is at least exponential and no constructions smaller than triply exponential are known [16].

Temporal extensions of *DL-Lite* [9] have been considered in the context of conceptual modeling [2,3,4], where the focus lies on checking concept satisfiability. OBDA, the second major use case of *DL-Lite*, with query answering as the most important reasoning problem [8], has not yet been studied in a temporal setting. Investigations of temporal query languages based on a combination of DL queries such as conjunctive queries (CQs) and temporal logics such as LTL have started only quite recently. In [14], a framework is developed that combines the two without much interference. The algorithm for query answering in this setting is an LTL-satisfiability test using a sub-procedure to answer (atemporal) CQs. In [5], a similar query language, a combination of LTL and CQs over the DL $\mathcal{ALC}$, is proposed. Its temporal component, however, is allowed to influence the DL queries via the notion of rigid names. The complexity increases depending on whether only rigid concept names or also rigid role names are allowed. Additionally, the latter paper also studies the so-called data complexity, where the complexity is measured only w.r.t. the size of the sensor data.

In this paper, we follow an approach suggested in [14] to combine the first-order rewriting techniques for atemporal query answering in logics of the *DL-Lite* family with a temporal component. The main idea is to use optimized database techniques to answer the actual queries. However, the existing techniques for answering temporal queries over temporal databases do not perfectly suit our purposes. In [12], the authors describe a temporal extension of the SQL query language that can answer temporal queries over a complete temporal database. However, in our setting the database containing all previous observations may grow huge very fast, but not all past observations are relevant for a particular query. In [11], an approach is described that reduces the amount of space needed; but the query language considered there allows only for past operators. In addition to describing how these approaches can be applied to our problem, we propose a new algorithm that extends the one from [11] and can also deal with future operators. An extended version of this paper has been accepted for publication in [6]. The formal proofs of our results can be found in [7].

2 Preliminaries

We first describe the DL component, and then the temporal component of our query language. The *DL-Lite* family consists of various DLs that are tailored towards conceptual modeling and allow to realize query answering using classical database techniques. We only consider *DL-Lite*$_{core}$ as a prototypical example.

Definition 1. *Let* $\mathsf{N_C}$, $\mathsf{N_R}$, *and* $\mathsf{N_I}$ *be non-empty, pairwise disjoint sets of* concept, role, *and* individual names, *respectively. A* role expression *is either a role name* $P_1 \in \mathsf{N_R}$ *or an* inverse role P_2^- *with* $P_2 \in \mathsf{N_R}$. *A* basic concept *is of the form* A *or* $\exists R$, *where* $A \in \mathsf{N_C}$ *and* R *is a role expression. A* general concept *is of the form* B *or* $\neg B$, *where* B *is a basic concept.*

A concept inclusion *is of the form* $B \sqsubseteq C$, *where* B *is a basic concept and* C *is a general concept. An* assertion *is of the form* $A(a)$ *or* $P(a, b)$, *where* $A \in \mathsf{N_C}$, $P \in \mathsf{N_R}$, *and* $a, b \in \mathsf{N_I}$. *A* TBox *(or* ontology*) is a finite set of concept inclusions, and an* ABox *is finite set of assertions.*

The semantics of *DL-Lite*$_{core}$ is defined through the notion of an interpretation.

Definition 2. *An* interpretation *is a pair* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, *where* $\Delta^{\mathcal{I}}$ *is a non-empty set (called* domain*) and* $\cdot^{\mathcal{I}}$ *is a function that assigns to every* $A \in \mathsf{N_C}$ *a set* $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, *to every* $P \in \mathsf{N_R}$ *a binary relation* $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, *and to every* $a \in \mathsf{N_I}$ *an element* $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. *This function is extended to role expressions, basic concepts, and general concepts as follows:* $(P^-)^{\mathcal{I}} := \{(e, d) \mid (d, e) \in P^{\mathcal{I}}\}$, $(\exists R)^{\mathcal{I}} := \{d \mid$ *there is an* $e \in \Delta^{\mathcal{I}}$ *such that* $(d, e) \in R^{\mathcal{I}}\}$, *and* $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$.

$\mathcal{I}$ *is a* model *of* $B \sqsubseteq C$ *if* $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, *of* $A(a)$ *if* $a^{\mathcal{I}} \in A^{\mathcal{I}}$, *and of* $P(a, b)$ *if* $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$. *We write* $\mathcal{I} \models \mathcal{T}$ *if* $\mathcal{I}$ *is a model of all concept inclusions in the TBox* $\mathcal{T}$, *and* $\mathcal{I} \models \mathcal{A}$ *if* $\mathcal{I}$ *is a model of all assertions in the ABox* $\mathcal{A}$.

We assume that all interpretations $\mathcal{I}$ satisfy the *unique name assumption* (UNA), i.e. for all $a, b \in \mathsf{N_I}$ with $a \neq b$, we have $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

We now introduce the notion of temporal knowledge bases. Intuitively, they contain sensor data (ABoxes) for all previous time points, and a global TBox.

Definition 3. *A* temporal knowledge base *(TKB)* $\mathcal{K} = \langle (\mathcal{A}_i)_{0 \leq i \leq n}, \mathcal{T} \rangle$ *consists of a finite sequence of ABoxes* $\mathcal{A}_i$ *and a TBox* $\mathcal{T}$, *where the ABoxes* $\mathcal{A}_i$ *can only contain concept names that also occur in* $\mathcal{T}$. *Let* $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$ *be a sequence of interpretations* $\mathcal{I}_i = (\Delta, \cdot^{\mathcal{I}_i})$ *over a fixed non-empty domain* Δ. *Then* $\mathfrak{I}$ *is a* model *of* $\mathcal{K}$ *(written* $\mathfrak{I} \models \mathcal{K}$*) if* $\mathcal{I}_i \models \mathcal{A}_i$ *and* $\mathcal{I}_i \models \mathcal{T}$ *for all* i, $0 \leq i \leq n$.

Similar to what was done in [5,14], our temporal query language is based on conjunctive queries [1,10]. The main difference is that we do not allow negation, as also in *DL-Lite* arbitrary negation is not allowed. The reason is that the reductions in Section 3 do not work in the presence of negation.

Definition 4. *Let* $\mathsf{N_V}$ *be a set of variables. A* conjunctive query *(CQ) is of the form* $\phi = \exists y_1, \ldots, y_m.\psi$, *where* $y_1, \ldots, y_m \in \mathsf{N_V}$ *and* ψ *is a (possibly empty) finite conjunction of* atoms *of the form* $A(z)$ *for* $A \in \mathsf{N_C}$ *and* $z \in \mathsf{N_V} \cup \mathsf{N_I}$

*(*concept atom*); or* $r(z_1, z_2)$ *for* $r \in \mathsf{N_R}$ *and* $z_1, z_2 \in \mathsf{N_V} \cup \mathsf{N_I}$ *(*role atom*). The empty conjunction is denoted by* true.

Temporal conjunctive queries *(TCQs) are built from CQs as follows: each CQ is a TCQ, and if* ϕ_1 *and* ϕ_2 *are TCQs, then so are* $\phi_1 \wedge \phi_2$ *(conjunction),* $\phi_1 \vee \phi_2$ *(disjunction),* $\bigcirc\phi_1$ *(strong next),* $\bullet\phi_1$ *(weak next),* $\bigcirc^-\phi_1$ *(strong previous),* $\bullet^-\phi_1$ *(weak previous),* $\phi_1 \,\mathsf{U}\, \phi_2$ *(until), and* $\phi_1 \,\mathsf{S}\, \phi_2$ *(since).*

The symbols $\bigcirc^-$, $\bullet^-$, and S are called *past operators*, the symbols $\bigcirc$, $\bullet$, and U are *future operators*. All results also hold in the presence of the additional temporal operators $\Box$ (always), $\Box^-$ (always in the past), $\Diamond$ (eventually), and $\Diamond^-$ (some time in the past) [7], but we omit them here for space reasons.

We denote the set of variables occurring in a TCQ ϕ by $\mathsf{Var}(\phi)$, and the set of free variables in ϕ by $\mathsf{FVar}(\phi)$. A TCQ ϕ is called *Boolean* if $\mathsf{FVar}(\phi) = \emptyset$. We further denote by $\mathsf{Sub}(\phi)$ the set of all TCQs occurring as subqueries in ϕ (including ϕ itself). A *union of conjunctive queries* (UCQ) is a disjunction of CQs. For our purposes, it is sufficient to define the semantics for Boolean CQs and TCQs. As usual, it is given using the notion of a homomorphism [10].

Definition 5. *Let* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ *be an interpretation and* ψ *be a Boolean CQ. A mapping* $\pi \colon \mathsf{Var}(\psi) \cup \mathsf{N_I} \to \Delta$ *is a* homomorphism *of* ψ *into* $\mathcal{I}$ *if* $\pi(a) = a^{\mathcal{I}}$ *for all* $a \in \mathsf{N_I}$, $\pi(z) \in A^{\mathcal{I}}$ *for all concept atoms* $A(z)$ *in* ψ, *and* $(\pi(z_1), \pi(z_2)) \in r^{\mathcal{I}}$ *for all role atoms* $r(z_1, z_2)$ *in* ψ. *We say that* $\mathcal{I}$ *is a* model *of* ψ *(written* $\mathcal{I} \models \psi$*) if there is such a homomorphism.*

Let now ϕ *be a Boolean TCQ. For a sequence of interpretations* $\mathfrak{I} = (\mathcal{I}_i)_{0 \le i \le n}$ *and* i *with* $0 \le i \le n$, *we define* $\mathfrak{I}, i \models \phi$ *by induction on the structure of* ϕ*:*

$\mathfrak{I}, i \models \exists y_1, \dots, y_m.\psi$ *iff* $\mathcal{I}_i \models \exists y_1, \dots, y_m.\psi$
$\mathfrak{I}, i \models \phi_1 \wedge \phi_2$ *iff* $\mathfrak{I}, i \models \phi_1$ *and* $\mathfrak{I}, i \models \phi_2$
$\mathfrak{I}, i \models \phi_1 \vee \phi_2$ *iff* $\mathfrak{I}, i \models \phi_1$ *or* $\mathfrak{I}, i \models \phi_2$
$\mathfrak{I}, i \models \bigcirc\phi_1$ *iff* $i < n$ *and* $\mathfrak{I}, i+1 \models \phi_1$
$\mathfrak{I}, i \models \bullet\phi_1$ *iff* $i < n$ *implies* $\mathfrak{I}, i+1 \models \phi_1$
$\mathfrak{I}, i \models \bigcirc^-\phi_1$ *iff* $i > 0$ *and* $\mathfrak{I}, i-1 \models \phi_1$
$\mathfrak{I}, i \models \bullet^-\phi_1$ *iff* $i > 0$ *implies* $\mathfrak{I}, i-1 \models \phi_1$
$\mathfrak{I}, i \models \phi_1 \,\mathsf{U}\, \phi_2$ *iff there is some* k, $i \le k \le n$ *such that* $\mathfrak{I}, k \models \phi_2$ *and* $\mathfrak{I}, j \models \phi_1$ *for all* j, $i \le j < k$
$\mathfrak{I}, i \models \phi_1 \,\mathsf{S}\, \phi_2$ *iff there is some* k, $0 \le k \le i$ *such that* $\mathfrak{I}, k \models \phi_2$ *and* $\mathfrak{I}, j \models \phi_1$ *for all* j, $k < j \le i$.

Here we assume that there is no time point before 0 or after n, similar to the temporal semantics used for LTL in [20] or for temporal query languages for databases [11,15,18]. As in classical LTL, one can show that $\phi_1 \,\mathsf{S}\, \phi_2$ is equivalent to $\phi_2 \vee (\phi_1 \wedge \bigcirc^-(\phi_1 \,\mathsf{S}\, \phi_2))$, and a similar equivalence holds for U.

We are now ready to introduce the central reasoning problem of this paper, namely finding certain answers to TCQs.

Definition 6. *Let* ϕ *be a TCQ,* $\mathfrak{I} = (\mathcal{I}_i)_{0 \le i \le n}$ *a sequence of interpretations, and* $i \ge 0$. *The mapping* $\mathfrak{a} \colon \mathsf{FVar}(\phi) \to \mathsf{N_I}$ *is an* answer *to* ϕ *w.r.t.* $\mathfrak{I}$ *at time*

point i if $\mathfrak{I}, i \models \mathfrak{a}(\phi)$, where $\mathfrak{a}(\phi)$ denotes the Boolean TCQ that is obtained from ϕ by replacing the free variables according to $\mathfrak{a}$. Let further $\mathcal{K} = \langle (\mathcal{A}_i)_{0 \leq i \leq n}, \mathcal{T} \rangle$ be a TKB. A mapping $\mathfrak{a} \colon \mathsf{FVar}(\phi) \to \mathsf{N_I}$ is a certain answer *to ϕ w.r.t. $\mathcal{K}$ at time point i if for every $\mathfrak{J} \models \mathcal{K}$, we have $\mathfrak{J}, i \models \mathfrak{a}(\phi)$.*

The set of all answers to ϕ w.r.t. $\mathfrak{I}$ at time point i is denoted by $\mathsf{Ans}(\phi, \mathfrak{I}, i)$, and the set of all certain answers to ϕ w.r.t. $\mathcal{K}$ is denoted by $\mathsf{Cert}(\phi, \mathcal{K}, i)$. Recall that our main interest lies in finding answers to queries at the last time point, i.e. computing the sets $\mathsf{Ans}(\phi, \mathfrak{I}) := \mathsf{Ans}(\phi, \mathfrak{I}, n)$ or $\mathsf{Cert}(\phi, \mathcal{K}) := \mathsf{Cert}(\phi, \mathcal{K}, n)$.

We will sometimes use the abbreviation $\mathsf{false} := A(x) \wedge A'(x)$, where A, A' are new concept names for which we assume that the concept inclusion $A \sqsubseteq \neg A'$ is contained in the global TBox $\mathcal{T}$.

3 Answering Temporal Conjunctive Queries

For computing the set of certain answers for a *conjunctive query*, the *rewriting approach* [8] can be employed. It compiles the information contained in the TBox into the query and evaluates the query w.r.t. the ABox (viewed as database) using classical database techniques. A similar approach is possible for TCQs.

Definition 7. *For an ABox $\mathcal{A}$, the interpretation $\mathsf{DB}(\mathcal{A}) := (\mathsf{N_I}, \cdot^{\mathsf{DB}(\mathcal{A})})$ is defined as follows:*

- $a^{\mathsf{DB}(\mathcal{A})} := a$ *for all* $a \in \mathsf{N_I}$;
- $A^{\mathsf{DB}(\mathcal{A})} := \{a \mid A(a) \in \mathcal{A}\}$ *for all* $A \in \mathsf{N_C}$; *and*
- $P^{\mathsf{DB}(\mathcal{A})} := \{(a, b) \mid P(a, b) \in \mathcal{A}\}$ *for all* $P \in \mathsf{N_R}$.

As shown in [8], this interpretation is the smallest model of $\mathcal{A}$. In order to employ database techniques, we must assume $\mathsf{N_I}$ and $\mathsf{DB}(\mathcal{A})$ to be finite.

Proposition 8 ([8]). *Let ψ be a CQ, $\mathcal{A}$ be an ABox, and $\mathcal{T}$ be a TBox. There is a* canonical model $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ *of $\mathcal{A}$ and $\mathcal{T}$ and a UCQ $\psi^{\mathcal{T}}$ such that*

$$\mathsf{Cert}(\psi, \langle \mathcal{A}, \mathcal{T} \rangle) = \mathsf{Ans}(\psi, \mathcal{I}_{\mathcal{A},\mathcal{T}}) = \mathsf{Ans}(\psi^{\mathcal{T}}, \mathsf{DB}(\mathcal{A})).$$

We now use this proposition to show a similar result for TCQs. Let ϕ be a TCQ and $\mathcal{K} = \langle (\mathcal{A}_i)_{0 \leq i \leq n}, \mathcal{T} \rangle$ be a TKB. The TCQ $\phi^{\mathcal{T}}$ is obtained by replacing each CQ ψ occurring in ϕ by $\psi^{\mathcal{T}}$. Note that $\phi^{\mathcal{T}}$ is again a TCQ since $\psi^{\mathcal{T}}$ is always a UCQ. Let furthermore $\mathfrak{I}_{\mathcal{K}} := (\mathcal{I}_{\mathcal{A}_i,\mathcal{T}})_{0 \leq i \leq n}$ and $\mathsf{DB}(\mathcal{K}) := (\mathsf{DB}(\mathcal{A}_i))_{0 \leq i \leq n}$. The following theorem can be shown by a straightforward induction on the structure of ϕ.

Theorem 9. *For every TCQ ϕ, TKB $\mathcal{K} = \langle (\mathcal{A}_i)_{0 \leq i \leq n}, \mathcal{T} \rangle$, and $i \geq 0$, we have* $\mathsf{Cert}(\phi, \mathcal{K}, i) = \mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}}, i) = \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K}), i)$.

More importantly, for every TCQ ϕ and TKB $\mathcal{K} = \langle (\mathcal{A}_i)_{0 \leq i \leq n}, \mathcal{T} \rangle$, it holds that $\mathsf{Cert}(\phi, \mathcal{K}) = \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K}))$. It thus remains to show how to compute the set $\mathsf{Ans}(\phi, \mathfrak{I})$ for a TCQ ϕ and a sequence $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$ of interpretations

over a finite domain. A first possibility is to view $\mathfrak{I}$ as a temporal database and rewrite ϕ into an ATSQL query [12]. However, since our goal is to monitor processes that produce new data in very short time intervals, storing all the data for all previous time points is not feasible. Therefore, we describe two different approaches that reduce the amount of space necessary to compute $\mathsf{Ans}(\phi, \mathfrak{I})$. Since we are interested in the answers at the last time point, the idea is to keep only the past information necessary to answer the query ϕ.

In the first approach (Section 4), we rewrite ϕ into a TCQ ϕ' without future operators, employing a construction described in [13]. We then compute $\mathsf{Ans}(\phi', \mathfrak{I})$ using an algorithm described in [11,19] that uses a so-called *bounded history encoding*, which means that the space required by the algorithm is constant w.r.t. the number n of previous time points. Only the current state of the database and some auxiliary relations have to be stored.

In Section 5, we generalize the algorithm from [11] to directly deal with the future operators. The main difference is that we do not consider negation or arbitrary first-order queries. Unfortunately, the space required by this algorithm is in general unbounded w.r.t. n and thus does not constitute a bounded history encoding in the sense of [11,19]. However, it allows us to circumvent the non-elementary blow-up of the formula resulting from the reduction in [13].

4 Eliminating Future Operators

In this section, we rewrite a TCQ ϕ into an equivalent TCQ ϕ' that does not contain future operators, but may contain negation as in [11]. We then apply the algorithm described in [11] to iteratively compute the sets $\mathsf{Ans}(\phi', \mathfrak{I}, i)$. The reduction uses the separation theorem for propositional LTL [13]. However, this theorem cannot be applied directly since our temporal semantics differs from that in [13]. The only temporal operators in [13] are strict versions of U and S. Moreover, the semantics is defined w.r.t. bounded past and unbounded future.

Definition 10. *Let P be a set of* propositional variables. LTL-formulae *are built from P using the constructors $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\neg\phi_1$, $\phi_1 \,\mathsf{U}^< \phi_2$, and $\phi_1 \,\mathsf{S}^< \phi_2$. An* LTL-structure *is an infinite sequence $\mathfrak{J} = (w_i)_{i\geq 0}$ of* worlds *$w_i \subseteq P$, $i \geq 0$, and it* satisfies *an LTL-formula ϕ at $i \geq 0$ (written $\mathfrak{J}, i \models \phi$) if*

$$
\begin{array}{ll}
\mathfrak{J}, i \models p \quad \textit{for } p \in P & \textit{iff } p \in w_i \\
\mathfrak{J}, i \models \phi_1 \wedge \phi_2 & \textit{iff } \mathfrak{J}, i \models \phi_1 \textit{ and } \mathfrak{J}, i \models \phi_2 \\
\mathfrak{J}, i \models \phi_1 \vee \phi_2 & \textit{iff } \mathfrak{J}, i \models \phi_1 \textit{ or } \mathfrak{J}, i \models \phi_2 \\
\mathfrak{J}, i \models \neg\phi_1 & \textit{iff not } \mathfrak{J}, i \models \phi_1 \\
\mathfrak{J}, i \models \phi_1 \,\mathsf{U}^< \phi_2 & \textit{iff there is some } k > i \textit{ such that } \mathfrak{J}, k \models \phi_2 \\
 & \quad \textit{and } \mathfrak{J}, j \models \phi_1 \textit{ for all } j,\ i < j < k \\
\mathfrak{J}, i \models \phi_1 \,\mathsf{S}^< \phi_2 & \textit{iff there is some } k,\ 0 \leq k < i, \textit{ such that } \mathfrak{J}, k \models \phi_2 \\
 & \quad \textit{and } \mathfrak{J}, j \models \phi_1 \textit{ for all } j,\ k < j < i.
\end{array}
$$

We define the constants true and false by $p \vee \neg p$ and $p \wedge \neg p$, respectively, for an arbitrary $p \in P$. We also define $\mathsf{first} := \neg(\mathsf{true}\,\mathsf{S}^<\,\mathsf{true})$ with the semantics that $\mathfrak{J}, i \models \mathsf{first}$ iff $i = 0$, i.e. this formula is satisfied exactly at the first time point.

Let from now on ϕ be an arbitrary but fixed TCQ containing only the CQs $\alpha_1, \ldots, \alpha_m$. Let furthermore $\{p_1, \ldots, p_m, p\}$ be the set of propositional variables. For a finite sequence $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$ of interpretations, its *propositional abstraction* is the LTL-structure $\widehat{\mathfrak{I}} := (w_i)_{i \geq 0}$, where $w_i := \{p_j \mid \mathcal{I}_i \models \alpha_j\} \cup \{p\}$ if $0 \leq i \leq n$, and $w_i := \emptyset$ otherwise. We first transform ϕ into an LTL-formula f_ϕ that behaves similarly to ϕ w.r.t. the propositional abstractions of sequences of interpretations $\mathfrak{I}$. It is defined inductively on the structure of ϕ:

- $f_{\alpha_j} := p_j$ for a CQ α_j;
- $f_{\phi_1 \wedge \phi_2} := f_{\phi_1} \wedge f_{\phi_2}$; $\quad f_{\phi_1 \vee \phi_2} := f_{\phi_1} \vee f_{\phi_2}$;
- $f_{\bigcirc \phi_1} := \mathsf{false}\, \mathsf{U}^< (f_{\phi_1} \wedge p)$; $\quad f_{\bigcirc^- \phi_1} := \mathsf{false}\, \mathsf{S}^< f_{\phi_1}$;
- $f_{\bullet \phi_1} := \mathsf{false}\, \mathsf{U}^< (f_{\phi_1} \vee \neg p)$; $\quad f_{\bullet^- \phi_1} := \mathsf{first} \vee \mathsf{false}\, \mathsf{S}^< f_{\phi_1}$;
- $f_{\phi_1\, \mathsf{U}\, \phi_2} := f_{\phi_2} \vee (f_{\phi_1} \wedge f_{\phi_1}\, \mathsf{U}^< (f_{\phi_2} \wedge p))$; and
- $f_{\phi_1\, \mathsf{S}\, \phi_2} := f_{\phi_2} \vee (f_{\phi_1} \wedge f_{\phi_1}\, \mathsf{S}^< f_{\phi_2})$.

We now use the separation theorem from [13] to transform f_ϕ into an equivalent LTL-formula f'_ϕ which is a Boolean combination of temporal subformulae that either contain only $\mathsf{S}^<$ operators or only $\mathsf{U}^<$ operators. In the proof of this theorem, subformulae of f_ϕ are copied and rearranged, but no additional propositional variables are introduced.

Since we are interested in evaluating ϕ (and thus f_ϕ and f'_ϕ) at n, we can now reduce f'_ϕ as follows: First, we replace all variables that are in the scope of a $\mathsf{U}^<$ by false. The reason for this is that such variables are only evaluated at time points after n, where all variables are false in all propositional abstractions. The resulting formula is then simplified using the simple equivalences $\neg\mathsf{true} \equiv \mathsf{false}$, $\neg\mathsf{false} \equiv \mathsf{true}$, $\mathsf{true} \wedge \psi \equiv \psi$, $\mathsf{true} \vee \psi \equiv \mathsf{true}$, $\mathsf{false} \wedge \psi \equiv \mathsf{false}$, $\mathsf{false} \vee \psi \equiv \psi$, $\psi\, \mathsf{U}^< \mathsf{false} \equiv \mathsf{false}$, $\mathsf{true}\, \mathsf{U}^< \mathsf{true} \equiv \mathsf{true}$, and $\mathsf{false}\, \mathsf{U}^< \mathsf{true} \equiv \mathsf{true}$.

This yields a formula f''_ϕ that does not contain any $\mathsf{U}^<$ operators and is equivalent to f'_ϕ at time point n in every LTL-structure of the form $\widehat{\mathfrak{I}}$. This formula is now translated back into a Boolean TCQ $\phi_{f''_\phi}$. Recall that the goal is to use the algorithm presented in [11], where negation is allowed in the query language. Furthermore, in that paper a slightly different operator S^* is used instead of S. The semantics of $\neg$ and S^*, as employed in [11], is as follows:

$$\begin{aligned} &\mathfrak{I}, i \models \neg\phi_1 \qquad \text{iff not } \mathfrak{I}, i \models \phi_1 \\ &\mathfrak{I}, i \models \phi_1\, \mathsf{S}^*\, \phi_2 \text{ iff there is some } k,\ 0 \leq k < i \text{ such that } \mathfrak{I}, k \models \phi_2 \\ &\qquad\qquad\qquad \text{and } \mathfrak{I}, j \models \phi_1 \text{ for all } j,\ k < j \leq i. \end{aligned}$$

In the following, we call any TCQ built using the operators $\wedge$, $\vee$, $\neg$, $\bigcirc^-$, and S^* a *Past-TCQ*, which is in particular a temporal query in the sense of [11]. We can now define the final translation recursively as follows:

- $\phi_{p_j} := \alpha_j$ for j, $1 \leq j \leq m$; $\quad \phi_p := \mathsf{true}$;
- $\phi_{f_1 \wedge f_2} := \phi_{f_1} \wedge \phi_{f_2}$; $\quad \phi_{f_1 \vee f_2} := \phi_{f_1} \vee \phi_{f_2}$; $\quad \phi_{\neg f_1} := \neg\phi_{f_1}$;
- $\phi_{f_1\, \mathsf{S}^<\, f_2} := \bigcirc^- (\phi_{f_2} \vee \phi_{f_1}\, \mathsf{S}^*\, \phi_{f_2})$.

In [7], we use the above constructions to show the following result.

Theorem 11. *For every Boolean TCQ ϕ there is a Boolean Past-TCQ ψ such that for all $\mathfrak{I} = (\mathcal{I}_i)_{0 \le i \le n}$, we have $\mathfrak{I}, n \models \phi$ iff $\mathfrak{I}, n \models \psi$.*

Recall that in the above construction the Boolean CQs α_j were only copied and rearranged inside the structure of ϕ. Thus, it is easy to see from Definition 6 that this result also holds for computing the sets of answers of non-Boolean TCQs.

Corollary 12. *For every TCQ ϕ there is a Past-TCQ ψ such that $\mathsf{FVar}(\psi) = \mathsf{FVar}(\phi)$ and for all $\mathfrak{I} = (\mathcal{I}_i)_{0 \le i \le n}$, we have $\mathsf{Ans}(\phi, \mathfrak{I}) = \mathsf{Ans}(\psi, \mathfrak{I})$.*

Let now $\mathfrak{I} = (\mathcal{I}_i)_{i \ge 0}$ be an *infinite* sequence of interpretations representing the observations over all time points. At each time point $i \ge 0$, we only have access to the finite prefix $\mathfrak{I}^{(i)} := (\mathcal{I}_j)_{0 \le j \le i}$ of $\mathfrak{I}$ of length $i + 1$. Let Δ be the shared domain of the interpretations in $\mathfrak{I}$. The algorithm from [11] works as follows on the TCQ ψ constructed in Corollary 12. On input $\mathcal{I}_0$, it computes a first-order interpretation $\mathcal{I}'_0$ of several auxiliary predicates. Intuitively, for each subformula ψ' of ψ beginning with a past operator, the algorithm stores the answers $\mathsf{Ans}(\psi', \mathfrak{I}^{(0)}) \subseteq \Delta^{\mathsf{FVar}(\psi')}$ for ψ' in a new relation $A^{\mathcal{I}'_0}_{\psi'} \subseteq \Delta^k$ of arity $k := |\mathsf{FVar}(\psi')|$. The set $\mathsf{Ans}(\psi, \mathfrak{I}^{(0)})$ can then easily be computed from $\mathcal{I}_0$ and $\mathcal{I}'_0$. Afterwards, the algorithm disregards $\mathcal{I}_0$ and keeps only the information computed in $\mathcal{I}'_0$. On input $\mathcal{I}_1$, it then updates $\mathcal{I}'_0$ to a new interpretation $\mathcal{I}'_1$, which allows it to compute $\mathsf{Ans}(\psi, \mathfrak{I}^{(1)})$, and so on.

The memory requirements of this algorithm are bounded by the maximal size of one pair $(\mathcal{I}_i, \mathcal{I}'_i)$, which is polynomial in the size of Δ, in the number of concept and role names, and in the number of past operators occurring in ψ, and exponential in the number of free variables occurring below past operators. However, the memory requirements do not depend on the length of the sequence of interpretations $\mathfrak{I}^{(i)}$ seen so far. This is called a *bounded history encoding* in [11].

The presented construction has several drawbacks. First, the translations from ϕ to f_ϕ and from f''_ϕ to $\phi_{f''_\phi} = \psi$ may duplicate subformulae, which can cause exponential blowups in the size of ϕ. This could be avoided by applying a reduction similar to the one for propositional LTL in [13] directly to the TCQ ϕ. However, since the reduction in [13] is already non-elementary in the size of the formula (basically one exponential for each $\mathsf{U}^<$ nested inside a $\mathsf{S}^<$, and vice versa), this is not much more efficient. The approach presented in this section is thus best suited for answering simple, small queries ϕ over large data sets $\mathfrak{I}$.

5 A New Algorithm

In this section, we present an algorithm that computes the set $\mathsf{Ans}(\phi, \mathfrak{I})$ without the need to eliminate the future operators beforehand, thereby avoiding the non-elementary blowup of the construction described in the previous section. However, the memory requirements of this new algorithm are not independent of the number of previous time points. From now on, let ϕ be a fixed TCQ and $\mathfrak{I} = (\mathcal{I}_i)_{i \ge 0}$ be a fixed infinite sequence of interpretations over the same finite domain Δ. For $i \ge 0$, we denote by $\mathfrak{I}^{(i)} := (\mathcal{I}_j)_{0 \le j \le i}$ the finite prefix of $\mathfrak{I}$ of

length $i+1$. Our algorithm iteratively computes the sets $\mathsf{Ans}(\phi, \mathfrak{I}^{(i)})$. It uses as data structure so-called *answer formulae*, which represent TCQs in which some parts have already been evaluated. In particular, they contain no more CQs, but rather sets of already computed answers to subqueries. Additionally, they may contain variables (different from those in $\mathsf{N_V}$) that serve as place-holders for subqueries that have to be evaluated at the next time point.

For ease of presentation, we assume in the following that $\mathsf{N_V}$ is finite and that answers are of the form $\mathfrak{a}\colon \mathsf{N_V} \to \Delta$ instead of $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \Delta$. In an implementation, one would of course already restrict the intermediate computations of answers for subqueries $\psi \in \mathsf{Sub}(\phi)$ to $\mathsf{FVar}(\psi)$. But then one has to be more careful when combining answers to different subqueries. Thus, when we talk about answers, we mean mappings $\mathfrak{a}\colon \mathsf{N_V} \to \Delta$, and in particular $\mathsf{Ans}(\ldots)$ refers to a set of such mappings, i.e. a subset of $\Delta^{\mathsf{N_V}}$.

Definition 13. *Let* $\mathsf{FSub}(\phi)$ *denote the set of all subqueries of* ϕ *of the form* $\bigcirc\psi_1$, $\bullet\psi_1$, *or* $\psi_1 \,\mathsf{U}\, \psi_2$. *For* $j \geq 0$, *we denote by* Var_j^{ϕ} *the set of all variables of the form* x_j^{ψ} *for* $\psi \in \mathsf{FSub}(\phi)$. *The set* AF_ϕ^i *of all* answer formulae for ϕ at $i \geq 0$ *is the smallest set satisfying the following conditions:*

- *Every set* $A \subseteq \Delta^{\mathsf{N_V}}$ *is an answer formula for* ϕ *at* i.
- *Every* $x_j^{\psi} \in \mathsf{Var}_j^{\phi}$ *with* $j \leq i$ *is an answer formula for* ϕ *at* i.
- *If* α_1 *and* α_2 *are answer formulae for* ϕ *at* i, *then so are* $\alpha_1 \cap \alpha_2$ *and* $\alpha_1 \cup \alpha_2$.

In order to evaluate these answer formulae, we introduce the notion of correctness. Intuitively, an answer formula α for ϕ at i is correct for i if we obtain the set $\mathsf{Ans}(\phi, \mathfrak{I}^{(i)})$ by replacing the variables x_j^{ψ} in α by appropriate sets of answers and evaluating $\cap$ and $\cup$ as set intersection and union, respectively.

Definition 14. *We define the function* $\mathsf{eval}^n\colon \mathsf{AF}_\phi^n \to 2^{\Delta^{\mathsf{N_V}}}$, $n \geq 0$, *as follows:*

- $\mathsf{eval}^n(A) := A$ *if* $A \subseteq \Delta^{\mathsf{N_V}}$;
- $\mathsf{eval}^n(x_j^{\psi}) := \begin{cases} \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, j+1) & \text{if } j < n \text{ and } \psi = \bigcirc\psi_1 \text{ or } \psi = \bullet\psi_1; \\ \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, j+1) & \text{if } j < n \text{ and } \psi = \psi_1 \,\mathsf{U}\, \psi_2; \\ \emptyset & \text{if } j = n \text{ and } \psi = \bigcirc\psi_1 \text{ or } \psi = \psi_1 \,\mathsf{U}\, \psi_2; \\ \Delta^{\mathsf{N_V}} & \text{if } j = n \text{ and } \psi = \bullet\psi_1; \end{cases}$
- $\mathsf{eval}^n(\alpha_1 \cap \alpha_2) := \mathsf{eval}^n(\alpha_1) \cap \mathsf{eval}^n(\alpha_2)$; *and*
- $\mathsf{eval}^n(\alpha_1 \cup \alpha_2) := \mathsf{eval}^n(\alpha_1) \cup \mathsf{eval}^n(\alpha_2)$.

We say that a mapping $\Phi\colon \mathsf{Sub}(\phi) \to \mathsf{AF}_\phi^i$ *is* correct for $i \geq 0$ *if for all* $n \geq i$ *and for all* $\psi \in \mathsf{Sub}(\phi)$, *we have* $\mathsf{eval}^n(\Phi(\psi)) = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)$.

In particular, if $\Phi\colon \mathsf{Sub}(\phi) \to \mathsf{AF}_\phi^i$ is correct for i, then $\mathsf{eval}^i(\Phi(\phi)) = \mathsf{Ans}(\phi, \mathfrak{I}^{(i)})$, which is the set we want to compute. The algorithm works as follows. It first computes a mapping Φ_0 that is correct for 0, which is used to compute the next mapping Φ_1 when the interpretation $\mathcal{I}_1$ becomes available. This mapping is correct for 1 and can be used to compute the next mapping Φ_2, and so on. In each step, to compute Φ_{i+1}, we only need Φ_i and the interpretation $\mathcal{I}_{i+1}$.

Definition 15. *We recursively define the mapping* $\Phi_0 \colon \mathsf{Sub}(\phi) \to \mathsf{AF}^0_\phi$*:*

- $\Phi_0(\psi_1) := \mathsf{Ans}(\psi_1, \mathfrak{I}^{(0)})$ *if* ψ_1 *is a CQ;*
- $\Phi_0(\psi_1 \wedge \psi_2) := \Phi_0(\psi_1) \cap \Phi_0(\psi_2)$*;* $\quad \Phi_0(\psi_1 \vee \psi_2) := \Phi_0(\psi_1) \cup \Phi_0(\psi_2)$*;*
- $\Phi_0(\bigcirc \psi_1) := x_0^{\bigcirc \psi_1}$*;* $\quad \Phi_0(\bigcirc^- \psi_1) := \emptyset$*;*
- $\Phi_0(\bullet \psi_1) := x_0^{\bullet \psi_1}$*;* $\quad \Phi_0(\bullet^- \psi_1) := \Delta^{\mathsf{Nv}}$*;*
- $\Phi_0(\psi_1 \,\mathsf{U}\, \psi_2) := \Phi_0(\psi_2) \cup \big(\Phi_0(\psi_1) \cap x_0^{\psi_1 \,\mathsf{U}\, \psi_2}\big)$*;* $\quad \Phi_0(\psi_1 \,\mathsf{S}\, \psi_2) := \Phi_0(\psi_2)$*.*

We assume for this definition that CQs ψ_1 are answered using a different mechanism, e.g. by evaluating the first-order query ψ_1 over the "database" $\mathcal{I}_0$ [1].

Assume now that $\Phi_{i-1} \colon \mathsf{Sub}(\phi) \to \mathsf{AF}^{i-1}_\phi$ is a function containing only variables with index $i-1$. We proceed as follows to construct a new function that contains only variables with index i. We first define a function $\Phi^0_i \colon \mathsf{Sub}(\phi) \to \mathsf{AF}^i_\phi$ similarly to Definition 15 that may still contain variables with index $i-1$. We then iteratively replace these variables until only variables with index i are left.

Definition 16. *Let* $i > 0$ *and* $\Phi_{i-1} \colon \mathsf{Sub}(\phi) \to \mathsf{AF}^{i-1}_\phi$*. We recursively define the mapping* $\Phi^0_i \colon \mathsf{Sub}(\phi) \to \mathsf{AF}^i_\phi$ *as follows:*

- $\Phi^0_i(\psi_1) := \mathsf{Ans}(\psi_1, \mathfrak{I}^{(i)})$ *if* ψ_1 *is a CQ;*
- $\Phi^0_i(\psi_1 \wedge \psi_2) := \Phi^0_i(\psi_1) \cap \Phi^0_i(\psi_2)$*;* $\quad \Phi^0_i(\psi_1 \vee \psi_2) := \Phi^0_i(\psi_1) \cup \Phi^0_i(\psi_2)$
- $\Phi^0_i(\bigcirc \psi_1) := x_i^{\bigcirc \psi_1}$*;* $\quad \Phi^0_i(\bigcirc^- \psi_1) := \Phi_{i-1}(\psi_1)$*;*
- $\Phi^0_i(\bullet \psi_1) := x_i^{\bullet \psi_1}$*;* $\quad \Phi^0_i(\bullet^- \psi_1) := \Phi_{i-1}(\psi_1)$*;*
- $\Phi^0_i(\psi_1 \,\mathsf{U}\, \psi_2) := \Phi^0_i(\psi_2) \cup \big(\Phi^0_i(\psi_1) \cap x_i^{\psi_1 \,\mathsf{U}\, \psi_2}\big)$*; and*
- $\Phi^0_i(\psi_1 \,\mathsf{S}\, \psi_2) := \Phi^0_i(\psi_2) \cup \big(\Phi^0_i(\psi_1) \cap \Phi_{i-1}(\psi_1 \,\mathsf{S}\, \psi_2)\big)$*.*

In order to remove the "old" variables with index $i-1$, we now substitute them appropriately. For example, since $x_{i-1}^{\bigcirc \psi}$ is a place-holder for the answers to ψ w.r.t. $\mathfrak{I}^{(n)}$ at i, we can now replace it by $\Phi^0_i(\psi)$. However, this formula may itself contain another old variable, and thus we have to be careful about the order in which we do these substitutions. Since each $\Phi^0_i(\psi)$ can contain only variables that refer to subqueries of ψ, by replacing the variables for "smaller" subqueries first, we ensure that all variables are eliminated. The details of this construction can be found in [7]. We obtain a mapping $\Phi_i \colon \mathsf{Sub}(\phi) \to \mathsf{AF}^i_\phi$ that is correct for i.

Lemma 17. *For each* $i \geq 0$*, the mapping* Φ_i *is correct for* i*.*

Consider now the algorithm, which, on input ϕ and $\mathfrak{I}$, computes the mappings Φ_i as described above, and outputs $\mathsf{eval}^i(\Phi_i(\phi))$ for each $i \geq 0$. The following is a trivial consequence of the correctness of these mappings.

Theorem 18. *Given a TCQ* ϕ *and an infinite sequence* $\mathfrak{I} = (\mathcal{I}_i)_{i \geq 0}$ *of interpretations, the algorithm outputs* $\mathsf{Ans}(\phi, \mathfrak{I}^{(i)})$ *for each* $i \geq 0$*.*

It is now easy to compute the sets $\mathsf{eval}^i(\Phi_i(\phi)) = \mathsf{Ans}(\phi, \mathfrak{I}^{(i)})$ for $i \geq 0$ since each of the variables x_i^ψ in $\Phi_i(\phi)$ simply has to be replaced by either $\emptyset$ or Δ^{Nv} (see Definition 14). However, as mentioned earlier, the size of the formula $\Phi_i(\phi)$ may

depend exponentially on the length i of the current sequence of interpretations. For example, the answer formula $\Phi_1^0(\phi)$ for $\phi = C(x)\,\mathsf{S}(A(x)\,\mathsf{U}\,B(x))$ is

$$\Phi_1^0(\psi) \cup (C_1 \cap (B_0 \cup (A_0 \cap x_0^\psi))),$$

where $\psi := A(x)\,\mathsf{U}\,B(x)$, $\Phi_1^0(\psi) = B_1 \cup (A_1 \cap x_1^\psi)$, $A_0 := \mathsf{Ans}(A(x), \mathfrak{I}^{(0)})$, and similarly for the other CQs and time points. After replacing x_0^ψ by $\Phi_1^0(\psi)$, the variable x_1^ψ occurs twice in $\Phi_1(\phi)$. In general, $\Phi_i(\phi)$ will contain 2^i occurrences of the variable x_i^ψ. However, applying equivalences like the associativity, commutativity, distributivity, and absorption laws for $\cap$ and $\cup$ does not affect the semantics of an answer formula (which is given by eval), and hence

$$\begin{aligned}
\Phi_1(\phi) &\equiv \Phi_1^0(\psi) \cup (C_1 \cap (B_0 \cup (A_0 \cap \Phi_1^0(\psi)))) \\
&\equiv \Phi_1^0(\psi) \cup (C_1 \cap B_0) \cup (C_1 \cap A_0 \cap \Phi_1^0(\psi)) \\
&\equiv \Phi_1^0(\psi) \cup (C_1 \cap B_0) \\
&\equiv ((C_1 \cap B_0) \cup B_1) \cup (A_1 \cap x_1^\psi)
\end{aligned}$$

The resulting formula contains x_1^ψ only once. In general, the formula $\Phi_i(\phi)$ is equivalent to $D_i \cup (A_i \cap x_i^\psi)$, where $D_0 := B_0$ and $D_i := (C_i \cap D_{i-1}) \cup B_i$ for all $i > 0$. Thus, the algorithm only has to store the two sets $A_i, D_i \subseteq \Delta^{\mathsf{N_V}}$ at each time point, i.e. we achieve a bounded history encoding, as in [11]. If the formula ϕ contains no future operators, then the answer formulae contain no variables and can always be fully evaluated to a subset of $\Delta^{\mathsf{N_V}}$. In this case, our algorithm can be seen as a variant of the one from [11] for less expressive queries.

The example demonstrates that it is important that the computed answer formulae are simplified at each step, while preserving their semantics under eval. However, this does not guarantee a bounded history encoding as in [11].

6 Conclusions

We have introduced the reasoning task of *temporal OBDA* over *DL-Lite* knowledge bases and shown how to reduce this task to answering queries over temporal databases, similar to what was done for the atemporal case [8]. We then presented three approaches to solve the latter problem. The first involves storing the whole history of the database and re-evaluating the query at each time point using a temporal database query language like ATSQL [12].

The second approach works by eliminating the future operators and evaluating the resulting query using the algorithm of [11], which achieves a bounded history encoding. Although independent of the length of the history, this involves a non-elementary blow-up in the size of the query. Finally, we presented an algorithm that works directly with the future operators. We showed that the algorithm computes exactly the desired answers, but its space requirements are in general not independent of the length of the history. In future work, we will try to achieve a bounded history encoding for certain classes of TCQs, and compare the performance of all three approaches on temporal databases.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Artale, A., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: *DL-Lite* with temporalised concepts, rigid axioms and roles. In: Ghilardi, S., Sebastiani, R. (eds.) Proc. of the 6th Int. Symp. on Frontiers of Combining Systems (FroCoS'09). Lecture Notes in Computer Science, vol. 5749, pp. 133–148. Springer-Verlag (2009)
3. Artale, A., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: Temporal conceptual modelling with DL-Lite. In: Proc. of the 2010 Int. Workshop on Description Logics (DL'10). CEUR-WS, vol. 573 (2010)
4. Artale, A., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: A cookbook for temporal conceptual data modelling with description logics. CoRR abs/1209.5571 (2012), `http://arxiv.org/abs/1209.5571`
5. Baader, F., Borgwardt, S., Lippmann, M.: Temporalizing ontology-based data access. In: Bonacina, M.P. (ed.) Proc. of the 24th Int. Conf. on Automated Deduction (CADE'13). Lecture Notes in Artificial Intelligence, vol. 7898, pp. 330–344. Springer-Verlag (2013)
6. Borgwardt, S., Lippmann, M., Thost, V.: Temporal query answering in the description logic *DL-Lite*. In: Fontaine, P., Ringeissen, C., Schmidt, R.A. (eds.) Proc. of the 9th Int. Symp. on Frontiers of Combining Systems (FroCoS'13). Lecture Notes in Computer Science, Springer-Verlag, Nancy, France (2013), to appear.
7. Borgwardt, S., Lippmann, M., Thost, V.: Temporal query answering w.r.t. *DL-Lite*-ontologies. LTCS-Report 13-05, Chair of Automata Theory, TU Dresden, Dresden, Germany (2013), see http://lat.inf.tu-dresden.de/research/reports.html.
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R.: Ontologies and databases: The DL-Lite approach. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.C., Schmidt, R.A. (eds.) Reasoning Web, 5th Int. Summer School 2009, Tutorial Lectures, Lecture Notes in Computer Science, vol. 5689, pp. 255–356. Springer-Verlag (2009)
9. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI'05). pp. 602–607. AAAI Press (2005)
10. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: Hopcroft, J.E., Friedman, E.P., Harrison, M.A. (eds.) Proc. of the 9th Annual ACM Symp. on Theory of Computing (STOC'77). pp. 77–90. ACM Press (1977)
11. Chomicki, J.: Efficient checking of temporal integrity constraints using bounded history encoding. ACM Transactions on Database Systems 20(2), 148–186 (1995)
12. Chomicki, J., Toman, D., Böhlen, M.H.: Querying ATSQL databases with temporal logic. ACM Transactions on Database Systems 26(2), 145–178 (2001)
13. Gabbay, D.: Declarative past and imperative future. In: Banieqbal, B., Barringer, H., Pnueli, A. (eds.) Proc. of the 1987 Coll. on Temporal Logic in Specification. Lecture Notes in Computer Science, vol. 398, pp. 409–448. Springer-Verlag (1989)
14. Gutiérrez-Basulto, V., Klarman, S.: Towards a unifying approach to representing and querying temporal data in description logics. In: Krötzsch, M., Straccia, U. (eds.) Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems (RR'12). Lecture Notes in Computer Science, vol. 7497, pp. 90–105. Springer-Verlag (2012)

15. Hülsmann, K., Saake, G.: Theoretical foundations of handling large substitution sets in temporal integrity monitoring. Acta Informatica 28(4), 365–407 (1991)
16. Laroussinie, F., Markey, N., Schnoebelen, P.: Temporal logic with forgettable past. In: Proc. of the 17th Annual IEEE Symp. on Logic in Computer Science (LICS'02). pp. 383–392. IEEE Press (2002)
17. Pnueli, A.: The temporal logic of programs. In: Proc. of the 18th Annual Symp. on Foundations of Computer Science (SFCS'77). pp. 46–57 (1977)
18. Saake, G., Lipeck, U.W.: Using finite-linear temporal logic for specifying database dynamics. In: Börger, E., Büning, H.K., Richter, M.M. (eds.) Proc. of the 2nd Workshop on Computer Science Logic (CSL'88), Lecture Notes in Computer Science, vol. 385, pp. 288–300. Springer-Verlag (1989)
19. Toman, D.: Logical data expiration. In: Chomicki, J., van der Meyden, R., Saake, G. (eds.) Logics for Emerging Applications of Databases, chap. 6, pp. 203–238. Springer-Verlag (2004)
20. Wilke, T.: Classifying discrete temporal properties. In: Proc. of the 16th Annual Symp. on Theoretical Aspects of Computer Science (STACS'99). Lecture Notes in Computer Science, vol. 1563, pp. 32–46. Springer-Verlag (1999)