TECHNISCHE UNIVERSITÄT DRESDEN

Technische Universität Dresden
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

Temporal Query Answering w.r.t. *DL-Lite*-Ontologies

Stefan Borgwardt Marcel Lippmann Veronika Thost

LTCS-Report 13-05

This revised version proves that the presented algorithm achieves a bounded history encoding.

Postal Address:
Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
01062 Dresden

http://lat.inf.tu-dresden.de

Visiting Address:
Nöthnitzer Str. 46
Dresden

Temporal Query Answering w.r.t. *DL-Lite*-Ontologies

Stefan Borgwardt Marcel Lippmann
Veronika Thost

Institute of Theoretical Computer Science
Technische Universität Dresden, Germany
{stefborg,lippmann,thost}@tcs.inf.tu-dresden.de

Abstract

Ontology-based data access (OBDA) generalizes query answering in relational databases. It allows to query a database by using the language of an ontology, abstracting from the actual relations of the database. For ontologies formulated in Description Logics of the *DL-Lite* family, OBDA can be realized by rewriting the query into a classical first-order query, e.g. an SQL query, by compiling the information of the ontology into the query. The query is then answered using classical database techniques.

In this report, we consider a temporal version of OBDA. We propose a temporal query language that combines a linear temporal logic with queries over $\mathit{DL\text{-}Lite}_{core}$-ontologies. This language is well-suited for expressing temporal properties of dynamical systems and is useful in context-aware applications that need to detect specific situations. Using a first-order rewriting approach, we transform our temporal queries into queries over a temporal database. We then present three approaches to answering the resulting queries, all having different advantages and drawbacks.

Contents

1 Introduction **3**

2 Preliminaries **5**

2.1 The *DL-Lite* Family 5

2.2 Temporal Conjunctive Queries 6

3 Answering Temporal Conjunctive Queries **9**

3.1 Computing the Answers 11

4 Eliminating Future Operators **12**

5 A New Algorithm **17**

5.1 The Initial Answer Formula 19

5.2 The Next Answer Formula 20

5.3 The Algorithm 22

6 Rigid Names **28**

7 Conclusions **33**

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) to be able 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 reacts accordingly (e.g. by increasing 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 [AHV95]. 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) [CDL+09], domain knowledge is encoded in ontologies using a description logic (DL). In this report, we consider logics of the *DL-Lite* family, which are light-weight DLs with a low complexity for many reasoning problems [CDL+09].

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 [Pnu77]. Usual temporal operators include *next* ($\bigcirc\phi$), which asserts that a property ϕ is true at the next point in time, *eventually* ($\Diamond\phi$), which asks for ϕ to be satisfied at some point in the future, and *always* ($\Box\phi$), which forces ϕ to be true at all time points in the future. 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 server it is running on 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, it is sufficient to have only past operators. We will even show that in our setting it is indeed 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. Indeed, for propositional LTL eliminating the past operators from a query results in a blowup that is at least exponential and no constructions of size less than triply exponential are known [LMS02].

Temporal extensions of *DL-Lite* [CGL+05] have been considered in the context of conceptual modeling [AKRZ09, AKRZ10, AKRZ12], where the focus lies on checking concept satisfiability instead of query answering. OBDA, the second major use case of *DL-Lite*, with query answering as the most important reasoning problem [CDL+09], 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 [GK12], 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 [BBL13], 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, which are names whose interpretation does not change over time. 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, i.e. the observations, but not w.r.t. the size of the query or the ontology.

In this report, we follow an approach suggested in [GK12] 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 [CTB01], 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 [Cho95], 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 [Cho95] and can also deal with future operators. All three approaches have different advantages and drawbacks.

Additionally, we show how the new algorithm can be extended to deal with rigid concept names for a specific subclass of queries. Unfortunately, there seems to be no simple way to adapt the algorithm to deal with rigid role names.

2 Preliminaries

In this report, we will consider temporal queries that are based on inexpressive DLs of the *DL-Lite* family [CDL+09]. We will first describe the DL component, and then the temporal component of this combination of logics.

2.1 The *DL-Lite* Family

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 2.1 (syntax of *DL-Lite*$_{core}$)**.** *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* $a, b \in \mathsf{N_I}$. *We call both concept inclusions and assertions* axioms.

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 2.2 (semantics of *DL-Lite*$_{core}$)**.** *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:

- $(P^-)^\mathcal{I} := \{(e, d) \mid (d, e) \in P^\mathcal{I}\}$
- $(\exists R)^\mathcal{I} := \{d \mid \textit{there is an } e \in \Delta^\mathcal{I} \textit{ such that } (d, e) \in R^\mathcal{I}\}$
- $(\neg C)^\mathcal{I} := \Delta^\mathcal{I} \setminus C^\mathcal{I}$

The interpretation $\mathcal{I}$ is a model *of the axiom α if:*

- $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ *if* $\alpha = B \sqsubseteq C$*;*
- $a^{\mathcal{I}} \in A^{\mathcal{I}}$ *if* $\alpha = A(a)$*; and*
- $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$ *if* $\alpha = P(a, b)$*.*

We write $\mathcal{I} \models \alpha$ if $\mathcal{I}$ is a model of the axiom α, and $\mathcal{I} \models \mathcal{T}$ ($\mathcal{I} \models \mathcal{A}$) if $\mathcal{I}$ is a model of all concept inclusions in the TBox $\mathcal{T}$ (all assertions in the ABox $\mathcal{A}$). An ABox $\mathcal{A}$ is consistent *(w.r.t. a TBox $\mathcal{T}$) if there is an interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \models \mathcal{T}$.*

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 $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

2.2 Temporal Conjunctive Queries

Before we introduce our temporal query language, we introduce the notion of temporal knowledge bases. Intuitively, they contain sensor data (ABoxes) for all previous time points, and a global TBox.

Definition 2.3. *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 (constant domain assumption). 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 [BBL13, GK12], our temporal query language is based on conjunctive queries [AHV95, CM77]. 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 2.4. *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),*
  - $\Box\phi_1$ *(always),* $\Box^-\phi_1$ *(always in the past),*
  - $\Diamond\phi_1$ *(eventually),* $\Diamond^-\phi_1$ *(history),*
  - $\phi_1 \,\mathsf{U}\, \phi_2$ *(until), and* $\phi_1 \,\mathsf{S}\, \phi_2$ *(since).*

The symbols $\bigcirc^-$, $\bullet^-$, $\Box^-$, $\Diamond^-$, and S are called *past operators*, the symbols $\bigcirc$, $\bullet$, $\Box$, $\Diamond$, and U are *future operators.*

We denote the set of individuals occurring in a TCQ ϕ by $\mathsf{Ind}(\phi)$, the set of variables occurring in ϕ by $\mathsf{Var}(\phi)$, the set of free variables in ϕ by $\mathsf{FVar}(\phi)$, and the set of atoms occurring in ϕ by $\mathsf{At}(\phi)$. 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 [CM77].

Definition 2.5. *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 $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$

and i with $0 \leq i \leq n$, we define $\mathfrak{I}, i \models \phi$ by induction on the structure of ϕ:

$\mathfrak{I}, i \models \exists y_1, \ldots, y_m.\psi$	*iff*	$\mathcal{I}_i \models \exists y_1, \ldots, y_m.\psi$
$\mathfrak{I}, i \models \phi_1 \wedge \phi_2$	*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* $\mathfrak{I}, i-1 \models \phi_1$
$\mathfrak{I}, i \models \Box\phi_1$	*iff*	*for all* k, $i \leq k \leq n$, *we have* $\mathfrak{I}, k \models \phi_1$
$\mathfrak{I}, i \models \Box^-\phi_1$	*iff*	*for all* k, $0 \leq k \leq i$, *we have* $\mathfrak{I}, k \models \phi_1$
$\mathfrak{I}, i \models \Diamond\phi_1$	*iff*	*there is some* k, $i \leq k \leq n$, *such that* $\mathfrak{I}, k \models \phi_1$
$\mathfrak{I}, i \models \Diamond^-\phi_1$	*iff*	*there is some* k, $0 \leq k \leq i$, *such that* $\mathfrak{I}, k \models \phi_1$
$\mathfrak{I}, i \models \phi_1 \mathsf{U} \phi_2$	*iff*	*there is some* k, $i \leq k \leq n$ *such that* $\mathfrak{I}, k \models \phi_2$ *and* $\mathfrak{I}, j \models \phi_1$ *for all* j, $i \leq j < k$
$\mathfrak{I}, i \models \phi_1 \mathsf{S} \phi_2$	*iff*	*there is some* k, $0 \leq k \leq i$ *such that* $\mathfrak{I}, k \models \phi_2$ *and* $\mathfrak{I}, j \models \phi_1$ *for all* j, $k < j \leq i$.

Given a TKB $\mathcal{K} = \langle (\mathcal{A}_i)_{0 \leq i \leq n}, \mathcal{T} \rangle$, we say that $\mathfrak{I}$ is a model *of ϕ w.r.t. $\mathcal{K}$ if $\mathfrak{I} \models \mathcal{K}$ and $\mathfrak{I}, n \models \phi$. We call ϕ satisfiable w.r.t. $\mathcal{K}$ if it has a model w.r.t. $\mathcal{K}$.*

Here we assume that there is no time point before 0 or after n, similar to the temporal semantics used for LTL in [Wil99] or for temporal query languages for databases [Cho95, HS91, SL89]. 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 similar equivalences hold for U, $\Box$, $\Box^-$, $\Diamond$, and $\Diamond^-$.

Proposition 2.6. *For $i > 0$, we have $\mathfrak{I}, i \models \phi_1 \mathsf{S} \phi_2$ iff*

- $\mathfrak{I}, i \models \phi_2$ *or*
- $\mathfrak{I}, i \models \phi_1$ *and* $\mathfrak{I}, i-1 \models \phi_1 \mathsf{S} \phi_2$.

Furthermore, $\mathfrak{I}, 0 \models \phi_1 \mathsf{S} \phi_2$ iff $\mathfrak{I}, 0 \models \phi_2$.

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

Definition 2.7. *Let ϕ be a TCQ, $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$ a sequence of interpretations, and $i \geq 0$. 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 $\mathfrak{I} \models \mathcal{K}$, we have $\mathfrak{I}, i \models \mathfrak{a}(\phi)$.*

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 $\mathsf{Cert}(\phi, \mathcal{K}) := \mathsf{Cert}(\phi, \mathcal{K}, n)$. If ϕ is a Boolean TCQ, then we have either $\mathsf{Cert}(\phi, \mathcal{K}, i) = \emptyset$ or $\mathsf{Cert}(\phi, \mathcal{K}, i) = \{\mathfrak{e}\}$, where $\mathfrak{e}$ denotes the empty function.

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* [CDL+09] 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 3.1. *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 [CDL+09], 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 3.2 ([CDL+09])**.** *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*

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

Note that the size of $\psi^{\mathcal{T}}$ is polynomial in the sizes of ψ and $\mathcal{T}$. 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 3.3. *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 $\mathsf{Cert}(\phi, \mathcal{K}, i) = \mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}}, i) = \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K}), i)$.*

Proof. We prove first $\mathsf{Cert}(\phi, \mathcal{K}, i) \subseteq \mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}}, i)$. Take $\mathfrak{a} \in \mathsf{Cert}(\phi, \mathcal{K}, i)$. Then we have for every $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$ with $\mathfrak{I} \models \mathcal{K}$, we have $\mathfrak{I}, i \models \mathfrak{a}(\phi)$. Note that the canonical models $\mathcal{I}_{\mathcal{A}_i, \mathcal{T}}$, as constructed in [CDL+09], are all countable, and thus we can assume without loss of generality that they are defined over the same domain. Thus, we have in particular that $\mathfrak{I}_{\mathcal{K}}, i \models \mathfrak{a}(\phi)$, which is equivalent to $\mathfrak{a} \in \mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}}, i)$.

It is left to prove the following two claims:

(1) $\mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}}, i) \subseteq \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K}), i)$, and

(2) $\mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K}), i) \subseteq \mathsf{Cert}(\phi, \mathcal{K}, i)$.

We show this by induction on the structure of ϕ.

For the base case, consider a ϕ of the form $\exists y_1, \ldots, y_m.\psi$. For (1), take an $\mathfrak{a} \in \mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}}, i)$. Then, we have $\mathfrak{I}_{\mathcal{K}}, i \models \mathfrak{a}(\phi)$. Since ϕ is a CQ, the semantics yields $\mathcal{I}_{\mathcal{A}_i, \mathcal{T}} \models \mathfrak{a}(\phi)$, which is equivalent to $\mathfrak{a} \in \mathsf{Ans}(\phi, \mathcal{I}_{\mathcal{A}_i, \mathcal{T}})$. By Proposition 3.2, this yields $\mathfrak{a} \in \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{A}_i))$. Again since ϕ is a CQ, we have that $\phi^{\mathcal{T}}$ is a UCQ, and thus the definition of Ans yields that $\mathfrak{a} \in \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K}), i)$.

For (2), take $\mathfrak{a} \in \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K}), i)$. Since ϕ is a CQ, the definitions of $\phi^{\mathcal{T}}$ and Ans yield that $\mathfrak{a} \in \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{A}_i))$. By Proposition 3.2, we have that $\mathfrak{a} \in \mathsf{Cert}(\phi, \langle \mathcal{A}_i, \mathcal{T} \rangle)$. This means that for every $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}_i$ and $\mathcal{I} \models \mathcal{T}$, we have that $\mathcal{I} \models \mathfrak{a}(\phi)$. Hence, we have also that for every $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$ with $\mathfrak{I} \models \mathcal{K}$, we have $\mathcal{I}_i \models \mathfrak{a}(\phi)$. Since ϕ is a CQ, this yields for every $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$ with $\mathfrak{I} \models \mathcal{K}$, we have $\mathfrak{I}, i \models \mathfrak{a}(\phi)$, and thus that $\mathfrak{a} \in \mathsf{Cert}(\phi, \mathcal{K}, i)$.

Let now, for the first inductive case, ϕ be of the form $\phi_1 \wedge \phi_2$. For (1), assume that $\mathfrak{I}_{\mathcal{K}}, i \models \mathfrak{a}(\phi) = \mathfrak{a}(\phi_1) \wedge \mathfrak{a}(\phi_2)$ holds. This yields that $\mathfrak{I}_{\mathcal{K}}, i \models \mathfrak{a}(\phi_1)$ and $\mathfrak{I}_{\mathcal{K}}, i \models \mathfrak{a}(\phi_2)$. By the induction hypothesis, we have $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi_1^{\mathcal{T}})$ and $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi_2^{\mathcal{T}})$. Hence $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi_1^{\mathcal{T}} \wedge \phi_2^{\mathcal{T}})$, and thus by definition of $\phi^{\mathcal{T}}$ we get $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi^{\mathcal{T}})$.

For (2), assume that $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi^{\mathcal{T}})$ holds. The definition of $\phi^{\mathcal{T}}$ yields that $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi_1^{\mathcal{T}}) \wedge \mathfrak{a}(\phi_2^{\mathcal{T}})$. Hence, we have $\mathfrak{a} \in \mathsf{Cert}(\phi_1, \mathcal{K}, i)$ and $\mathfrak{a} \in \mathsf{Cert}(\phi_2, \mathcal{K}, i)$ by the induction hypothesis. Thus, for every $\mathfrak{I} \models \mathcal{K}$ it holds that $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ and $\mathfrak{I}, i \models \mathfrak{a}(\phi_2)$. This is equivalent to $\mathfrak{a} \in \mathsf{Cert}(\phi_1 \wedge \phi_2, \mathcal{K}, i)$.

Let now ϕ be of the form $\bigcirc \phi_1$. For (1), take $\mathfrak{I}_{\mathcal{K}}, i \models \mathfrak{a}(\bigcirc \phi_1) = \bigcirc \mathfrak{a}(\phi_1)$. Thus, we have $i < n$ and $\mathfrak{I}_{\mathcal{K}}, i+1 \models \mathfrak{a}(\phi_1)$. By the induction hypothesis, we get $\mathsf{DB}(\mathcal{K}), i+1 \models \mathfrak{a}(\phi_1^{\mathcal{T}})$. Since $i < n$, this implies that $\mathsf{DB}(\mathcal{K}), i \models \bigcirc \mathfrak{a}(\phi_1^{\mathcal{T}})$, which is equivalent to $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi^{\mathcal{T}})$ by definition of $\phi^{\mathcal{T}}$.

For (2), let $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi^{\mathcal{T}})$. The definition of $\phi^{\mathcal{T}}$ yields that $\mathsf{DB}(\mathcal{K}), i \models \bigcirc \mathfrak{a}(\phi_1^{\mathcal{T}})$. Hence, we have $i < n$ and $\mathsf{DB}(\mathcal{K}), i+1 \models \mathfrak{a}(\phi_1^{\mathcal{T}})$, which implies $\mathfrak{a} \in \mathsf{Cert}(\phi_1, \mathcal{K}, i+1)$ by the induction hypothesis. Since $i < n$, this means that for every $\mathfrak{I} \models \mathcal{K}$ we have $\mathfrak{I}, i \models \bigcirc \mathfrak{a}(\phi_1)$, which shows that $\mathfrak{a} \in \mathsf{Cert}(\bigcirc \phi_1, \mathcal{K}, i)$.

For the next inductive case, let ϕ be of the form $\phi_1 \mathsf{U} \phi_2$. For (1), assume that $\mathfrak{I}_{\mathcal{K}}, i \models \mathfrak{a}(\phi_1 \mathsf{U} \phi_2) = \mathfrak{a}(\phi_1) \mathsf{U} \mathfrak{a}(\phi_2)$, and thus there is a k, $i \leq k \leq n$, such that $\mathfrak{I}_{\mathcal{K}}, k \models \mathfrak{a}(\phi_2)$ and $\mathfrak{I}_{\mathcal{K}}, j \models \mathfrak{a}(\phi_1)$ for all j, $i \leq j < k$. By the induction hypothesis, we have $\mathsf{DB}(\mathcal{K}), k \models \mathfrak{a}(\phi_2^{\mathcal{T}})$ and $\mathsf{DB}(\mathcal{K}), j \models \mathfrak{a}(\phi_1^{\mathcal{T}})$ for all j, $i \leq j < k$. The definitions of $\models$ and $\phi^{\mathcal{T}}$ yield that $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi^{\mathcal{T}})$.

For (2), assume that $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi^{\mathcal{T}})$. The definition of $\phi^{\mathcal{T}}$ yields $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi_1^{\mathcal{T}}) \mathsf{U} \mathfrak{a}(\phi_2^{\mathcal{T}})$. Hence, there is a k, $i \leq k \leq n$, such that $\mathfrak{I}_{\mathcal{K}}, k \models \mathfrak{a}(\phi_2)$ and $\mathfrak{I}_{\mathcal{K}}, j \models \mathfrak{a}(\phi_1)$ for all j, $i \leq j < k$. The induction hypothesis yields that $\mathfrak{a} \in \mathsf{Cert}(\phi_2, \mathcal{K}, k)$ and $\mathfrak{a} \in \mathsf{Cert}(\phi_1, \mathcal{K}, j)$ for all j, $i \leq j < k$. Thus, we have for every $\mathfrak{I} \models \mathcal{K}$ that $\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \mathsf{U} \mathfrak{a}(\phi_2) = \mathfrak{a}(\phi_1 \mathsf{U} \phi_2)$.

The remaining cases can be proven in a similar way. For example, the case of $\bullet\phi_1$ differs from $\bigcirc\phi_1$ only in the fact that if $i \geq n$, then the expressions $\mathfrak{I}_{\mathcal{K}}, i \models \mathfrak{a}(\phi)$ and $\mathsf{DB}(\mathcal{K}), i \models \mathfrak{a}(\phi^{\mathcal{T}})$ are trivially satisfied, instead of trivially false. The arguments for $\bigcirc^{-}\phi_1$ and $\bullet^{-}\phi_1$ can be obtained from those of $\bigcirc\phi_1$ and $\bullet\phi_1$ by replacing $i < n$ by $i > 0$ and $i + 1$ by $i - 1$, and similarly for $\phi_1 \mathsf{S} \phi_2$ and $\phi_1 \mathsf{U} \phi_2$. $\square$

We immediately get the following corollary.

Corollary 3.4. *For every TCQ ϕ and TKB $\mathcal{K} = \langle (\mathcal{A}_i)_{0 \leq i \leq n}, \mathcal{T} \rangle$, we have*

$$\mathsf{Cert}(\phi, \mathcal{K}) = \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K})).$$

3.1 Computing the Answers

It now 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 [CTB01]. 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 [Gab89]. We then compute $\mathsf{Ans}(\phi', \mathfrak{I})$ using an algorithm described in [Cho95, Tom04] 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 [Cho95] to directly deal with the future operators. The main difference is that we do not consider negation or

arbitrary first-order queries. We show that the space required by this algorithm is constant w.r.t. n and thus constitutes a bounded history encoding in the sense of [Cho95, Tom04]. This algorithm allows us to circumvent the non-elementary blow-up of the formula resulting from the reduction in [Gab89].

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 [Cho95]. We then apply the algorithm described in [Cho95] to iteratively compute the sets $\mathsf{Ans}(\phi', \mathfrak{I}, i)$. The reduction uses the separation theorem for propositional LTL [Gab89]. However, this theorem cannot be applied directly since our temporal semantics differs from that in [Gab89]. The only temporal operators in [Gab89] are strict versions of U and S. It is well-known that those operators can simulate $\bigcirc$ and $\bigcirc^-$. Moreover, the semantics is defined w.r.t. bounded past and unbounded future.

Definition 4.1. *Let P be a set of* propositional variables*.* LTL-formulae *are built from P using the constructors $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\neg\phi_1$, $\phi_1 \,\mathsf{U}^< \phi_2$ (*strict until*), and $\phi_1 \,\mathsf{S}^< \phi_2$ (*strict since*). An* LTL-structure *is an infinite sequence $\mathfrak{I} = (w_i)_{i\geq 0}$ of* worlds *$w_i \subseteq P$, $i \geq 0$, and it* satisfies *an LTL-formula ϕ at $i \geq 0$ (written $\mathfrak{I}, i \models \phi$) if*

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

As usual, 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 $\mathfrak{I}, i \models \mathsf{first}$ 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

$$
w_i := \begin{cases} \{p_j \mid \mathcal{I}_i \models \alpha_j\} \cup \{p\} & \text{if } 0 \leq i \leq n, \text{ and} \\ \emptyset & \text{otherwise.} \end{cases}
$$

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;
- $f_{\phi_1 \wedge \phi_2} := f_{\phi_1} \wedge f_{\phi_2}$; $\quad f_{\phi_1 \vee \phi_2} := f_{\phi_1} \vee f_{\phi_2}$;
- $f_{\bigcirc \phi_1} := \mathsf{false}\,\mathsf{U}^{<}(f_{\phi_1} \wedge p)$; $\quad f_{\bigcirc^- \phi_1} := \mathsf{false}\,\mathsf{S}^{<} f_{\phi_1}$;
- $f_{\bullet \phi_1} := \mathsf{false}\,\mathsf{U}^{<}(f_{\phi_1} \vee \neg p)$; $\quad f_{\bullet^- \phi_1} := \mathsf{first} \vee \mathsf{false}\,\mathsf{S}^{<} f_{\phi_1}$;
- $f_{\Box \phi_1} := f_{\phi_1} \wedge f_{\phi_1}\,\mathsf{U}^{<} \neg p$; $\quad f_{\Box^- \phi_1} := f_{\phi_1} \wedge f_{\phi_1}\,\mathsf{S}^{<}(\mathsf{first} \wedge f_{\phi_1})$;
- $f_{\Diamond \phi_1} := f_{\phi_1} \vee \mathsf{true}\,\mathsf{U}^{<}(f_{\phi_1} \wedge p)$; $\quad f_{\Diamond^- \phi_1} := f_{\phi_1} \vee \mathsf{true}\,\mathsf{S}^{<} f_{\phi_1}$;
- $f_{\phi_1 \,\mathsf{U}\, \phi_2} := f_{\phi_2} \vee (f_{\phi_1} \wedge f_{\phi_1}\,\mathsf{U}^{<}(f_{\phi_2} \wedge p))$; 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 can now prove the following lemma, which implies the above claim.

Lemma 4.2. *For all $\mathfrak{I} = (\mathcal{I}_i)_{0 \le i \le n}$ and all i, $0 \le i \le n$, we have $\mathfrak{I}, i \models \phi$ iff $\widehat{\mathfrak{I}}, i \models f_\phi$.*

Proof. We prove this lemma by induction on the structure of ϕ.

For the base case of a CQ $\phi = \alpha_j$, we have $\mathfrak{I}, i \models \alpha_j$ iff $\mathcal{I}_i \models \alpha_j$ iff $p_j \in w_i$ iff $\widehat{\mathfrak{I}}, i \models f_{\alpha_j}$.

For the case $\phi = \phi_1 \wedge \phi_2$, we have $\mathfrak{I}, i \models \phi_1 \wedge \phi_2$ iff $\mathfrak{I}, i \models \phi_1$ and $\mathfrak{I}, i \models \phi_2$ iff $\widehat{\mathfrak{I}}, i \models f_{\phi_1}$ and $\widehat{\mathfrak{I}}, i \models f_{\phi_2}$ iff $\widehat{\mathfrak{I}}, i \models f_{\phi_1} \wedge f_{\phi_2}$.

For the case $\phi = \bigcirc \phi_1$, we have $\mathfrak{I}, i \models \bigcirc \phi_1$ iff $i < n$ and $\mathfrak{I}, i+1 \models \phi_1$ iff $\widehat{\mathfrak{I}}, i+1 \models p$ and $\widehat{\mathfrak{I}}, i+1 \models f_{\phi_1}$ iff $\widehat{\mathfrak{I}}, i+1 \models f_{\phi_1} \wedge p$ iff $\widehat{\mathfrak{I}}, i \models \mathsf{false}\,\mathsf{U}^{<}(f_{\phi_1} \wedge p)$.

For the case $\phi = \bullet^- \phi_1$, we have $\mathfrak{I}, i \models \bullet^- \phi_1$ iff $i > 0$ implies $\mathfrak{I}, i-1 \models \phi_1$ iff $i = 0$ or $i > 0$ and $\mathfrak{I}, i-1 \models \phi_1$ iff $\widehat{\mathfrak{I}}, i \models \mathsf{first}$ or $i > 0$ and $\widehat{\mathfrak{I}}, i-1 \models f_{\phi_1}$ iff $\widehat{\mathfrak{I}}, i \models \mathsf{first} \vee \mathsf{false}\,\mathsf{S}^{<} f_{\phi_1}$.

For the case $\phi = \phi_1 \,\mathsf{U}\, \phi_2$, we have:

$\mathfrak{I}, i \models \phi_1 \,\mathsf{U}\, \phi_2$

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, $i \le j < k$

iff there is some k, $i \le k \le n$, such that $\widehat{\mathfrak{I}}, k \models f_{\phi_2}$ and $\widehat{\mathfrak{I}}, j \models f_{\phi_1}$ for all j, $i \le j < k$

iff there is some $k \geq i$, such that $\widehat{\mathfrak{I}}, k \models p$ and $\widehat{\mathfrak{I}}, k \models f_{\phi_2}$ and $\widehat{\mathfrak{I}}, j \models f_{\phi_1}$ for all j, $i \leq j < k$

iff $\widehat{\mathfrak{I}}, i \models f_{\phi_2}$ *or* there is some $k > i$, such that $\widehat{\mathfrak{I}}, k \models p$ and $\widehat{\mathfrak{I}}, k \models f_{\phi_2}$ and $\widehat{\mathfrak{I}}, j \models f_{\phi_1}$ for all j, $i \leq j < k$

iff $\widehat{\mathfrak{I}}, i \models f_{\phi_2}$ *or* there is some $k > i$, such that $\widehat{\mathfrak{I}}, k \models f_{\phi_2} \wedge p$ and $\widehat{\mathfrak{I}}, j \models f_{\phi_1}$ for all j, $i \leq j < k$

iff $\widehat{\mathfrak{I}}, i \models f_{\phi_2}$ *or* $\widehat{\mathfrak{I}}, i \models f_{\phi_1}$ *and* there is some $k > i$, such that $\widehat{\mathfrak{I}}, k \models f_{\phi_2} \wedge p$ and $\widehat{\mathfrak{I}}, j \models f_{\phi_1}$ for all j, $i < j < k$

iff $\widehat{\mathfrak{I}}, i \models f_{\phi_2} \vee (f_{\phi_1} \wedge f_{\phi_1} \mathsf{U}^{<} (f_{\phi_2} \wedge p))$.

All the other cases can be proved analogously. □

We now use the separation theorem from [Gab89] 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.

Proposition 4.3 ([Gab89])**.** *There is an LTL-formula f'_ϕ in which no $\mathsf{S}^{<}$ occurs in the scope of an $\mathsf{U}^{<}$ and no $\mathsf{U}^{<}$ occurs in the scope of an $\mathsf{S}^{<}$ such that for every LTL-structure $\mathfrak{I}$ and every $i \geq 0$, we have $\mathfrak{I}, i \models f_\phi$ iff $\mathfrak{I}, i \models f'_\phi$.*

Since we are interested in evaluating ϕ (and thus f_ϕ and f'_ϕ) at time point 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 following equivalences:

$$\begin{aligned} \mathsf{true} \wedge \psi &\equiv \psi & \neg\mathsf{true} &\equiv \mathsf{false} \\ \mathsf{true} \vee \psi &\equiv \mathsf{true} & \psi\, \mathsf{U}^{<}\, \mathsf{false} &\equiv \mathsf{false} \\ \mathsf{false} \wedge \psi &\equiv \mathsf{false} & \mathsf{true}\, \mathsf{U}^{<}\, \mathsf{true} &\equiv \mathsf{true} \\ \mathsf{false} \vee \psi &\equiv \psi & \mathsf{false}\, \mathsf{U}^{<}\, \mathsf{true} &\equiv \mathsf{true} \\ \neg\mathsf{false} &\equiv \mathsf{true} \end{aligned}$$

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 the LTL-structure of the $\widehat{\mathfrak{I}}$.

Lemma 4.4. *For all $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$, we have $\widehat{\mathfrak{I}}, n \models f'_\phi$ iff $\widehat{\mathfrak{I}}, n \models f''_\phi$.*

Proof. According to the semantics of $\mathsf{U}^<$, every propositional variable p_j occurring in the scope of any number of nested $\mathsf{U}^<$ operators in f'_ϕ is evaluated w.r.t. $\widehat{\mathfrak{I}}$ only at time points $n' > n$. Thus, all of these occurrences can be replaced by false without affecting the value of f'_ϕ under $\widehat{\mathfrak{I}}$ at time point n. Furthermore, the equivalences listed above are clearly valid at any time point, and thus also do not affect this value. □

We now translate the LTL-formula f''_ϕ without $\mathsf{U}^<$ back into a Boolean TCQ $\phi_{f''_\phi}$. Recall that the goal is to use the algorithm presented in [Cho95], 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 [Cho95], is as follows:

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

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 [Cho95]. We can now define the final translation recursively as follows:

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

As before, we have to show that the Boolean Past-TCQ $\phi_{f''_\phi}$ is satisfied by $\mathfrak{I}$ at time point i, $0 \leq i \leq n$, if and only if f''_ϕ is satisfied by $\widehat{\mathfrak{I}}$ at i.

Lemma 4.5. *For all $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$ and all i, $0 \leq i \leq n$, we have $\widehat{\mathfrak{I}}, i \models f''_\phi$ iff $\mathfrak{I}, i \models \phi_{f''_\phi}$.*

Proof. We prove this by induction on the structure of f''_ϕ.

For a propositional variable p_j, $1 \leq j \leq m$, we have $\widehat{\mathfrak{I}}, i \models p_j$ iff $\mathfrak{I}, i \models \alpha_j$ as in the proof of Lemma 4.2. For p, we have $\widehat{\mathfrak{I}}, i \models p$ iff $\mathfrak{I}, i \models \mathsf{true}$ since $p \in w_i$ holds for all i, $0 \leq i \leq n$.

For the Boolean operators $\wedge$, $\vee$, $\neg$, the claim follows similarly as in the proof of Lemma 4.2. It thus remains to show the claim for subformulae of the form $f_1 \,\mathsf{S}^<\, f_2$. We have

$$\widehat{\mathfrak{I}}, i \models f_1 \,\mathsf{S}^<\, f_2$$

iff there is some k, $0 \leq k < i$, such that $\widehat{\mathfrak{I}}, k \models f_2$ and $\widehat{\mathfrak{I}}, j \models f_1$ for all j, $k < j < i$

iff there is some k, $0 \leq k < i$, such that $\mathfrak{I}, k \models \phi_{f_2}$ and $\mathfrak{I}, j \models \phi_{f_1}$ for all j, $k < j < i$

iff $i > 0$ and $\mathfrak{I}, i-1 \models \phi_{f_2}$ *or* there is some k, $0 \leq k < i-1$ such that $\mathfrak{I}, k \models \phi_{f_2}$ and $\mathfrak{I}, j \models \phi_{f_1}$ for all j, $k < j \leq i-1$.

iff $i > 0$ and $\mathfrak{I}, i-1 \models \phi_{f_2}$ *or* $\mathfrak{I}, i-1 \models \phi_{f_1} \mathsf{S}^* \phi_{f_2}$

iff $\mathfrak{I}, i \models \bigcirc^-(\phi_{f_2} \vee \phi_{f_1} \mathsf{S}^* \phi_{f_2})$ □

From the previous lemmata and the separation theorem, we get the following result.

Theorem 4.6. *For every Boolean TCQ ϕ there is a Boolean Past-TCQ ψ such that for all $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$, 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 2.7 that this result also holds for computing the sets of answers of non-Boolean TCQs.

Corollary 4.7. *For every TCQ ϕ there is a Past-TCQ ψ such that $\mathsf{FVar}(\psi) = \mathsf{FVar}(\phi)$ and for all $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$, we have $\mathsf{Ans}(\phi, \mathfrak{I}) = \mathsf{Ans}(\psi, \mathfrak{I})$.*

This shows that we can compute $\mathsf{Ans}(\phi, \mathfrak{I})$ using the algorithm from [Cho95] as follows. The main advantage of this approach is that we can compute this set *iteratively* and such that the required memory is independent of the length of the sequence $\mathfrak{I}$. More formally, let $\mathfrak{I} = (\mathcal{I}_i)_{i \geq 0}$ be an *infinite* sequence of interpretations representing the observations over all time points. In our setting, these interpretations are generated from an infinite sequence of ABoxes that represent the observed sensor data using the construction of Section 3. At each time point $i \geq 0$, we only have access to the finite prefix $\mathfrak{I}^{(i)} := (\mathcal{I}_j)_{0 \leq j \leq i}$ of $\mathfrak{I}$ of length $i+1$. Let Δ be the shared domain of the interpretations in $\mathfrak{I}$.

The algorithm from [Cho95] works as follows on the TCQ ψ constructed in Corollary 4.7. 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 $A_{\psi'}^{\mathcal{I}'_0} \subseteq \Delta^k$ 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 [Cho95].

Note that a formal requirement for the correctness of the algorithm in [Cho95] is that ψ is *domain-independent*, which means that the answers to ψ at previous time points do not change if the domain is increased from the current time point to the next, e.g. by introducing new individuals. Otherwise, the answers to the past formulae at the current time point could not be so easily compiled into a single interpretation $\mathcal{I}_i$, but would have to be recomputed at each time point, and thus the algorithm would have to store the whole sequence $\mathfrak{I}^{(i)}$. However, since we are here only dealing with the constant, finite domain $\Delta := \mathsf{N_I}$ (see Section 3), we do not need to assume domain-independence of ψ.

The construction presented in this section 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 [Gab89] directly to the TCQ ϕ. However, since the reduction in [Gab89] 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 of non-elementary complexity in the size of the query, but does not depend on the number of previous time points. It 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. As the algorithm from [Cho95], it retains a data structure that can be used to compute the certain answers at the next time point once a new interpretation $\mathcal{I}_{n+1}$ becomes available. We can even show that the memory requirements of this new algorithm are independent of the number of previous time points. From now on, let ϕ be a fixed TCQ and $\mathfrak{I} = (\mathcal{I}_i)_{i\geq 0}$ be a fixed infinite sequence of interpretations over the same finite domain Δ. For $i \geq 0$, we denote by $\mathfrak{I}^{(i)} := (\mathcal{I}_j)_{0\leq j\leq i}$ the finite prefix of $\mathfrak{I}$ of length $i+1$. In the following, we describe an algorithm that iteratively computes the sets $\mathsf{Ans}(\phi, \mathfrak{I}^{(i)})$. For ease of presentation, in this section we do not consider the temporal operators $\Box$, $\Diamond$, $\Box^-$, and $\Diamond^-$. The constructions and arguments for these operators are similar to those for U and S.

The algorithm uses as data structure so-called *answer formulae*, which represents 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 $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \Delta$. After computing such a mapping, it can be restricted to $\mathsf{FVar}(\phi)$ to get the actual answer. 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 5.1. *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* $\alpha_1 \cup \alpha_2$.

Note that every answer formula at i is also an answer formula at $i+1$, i.e. we have $\mathsf{AF}_\phi^i \subseteq \mathsf{AF}_\phi^{i+1}$. 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 5.2. *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 $\Phi\colon \mathsf{Sub}(\phi) \to \mathsf{AF}^i_\phi$ 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 $\mathcal{I}_{i+1}$.

5.1 The Initial Answer Formula

We now construct Φ_0 and show that it is correct for 0.

Definition 5.3. *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;*
- $\Phi_0(\psi_1 \wedge \psi_2) := \Phi_0(\psi_1) \cap \Phi_0(\psi_2)$*;* $\quad \Phi_0(\psi_1 \vee \psi_2) := \Phi_0(\psi_1) \cup \Phi_0(\psi_2)$*;*
- $\Phi_0(\bigcirc\psi_1) := x_0^{\bigcirc\psi_1}$*;* $\quad \Phi_0(\bigcirc^-\psi_1) := \emptyset$*;*
- $\Phi_0(\bullet\psi_1) := x_0^{\bullet\psi_1}$*;* $\quad \Phi_0(\bullet^-\psi_1) := \Delta^{\mathsf{N_V}}$*;*
- $\Phi_0(\psi_1 \,\mathsf{U}\, \psi_2) := \Phi_0(\psi_2) \cup \left(\Phi_0(\psi_1) \cap x_0^{\psi_1 \,\mathsf{U}\, \psi_2}\right)$*;* $\quad \Phi_0(\psi_1 \,\mathsf{S}\, \psi_2) := \Phi_0(\psi_2)$.

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" $\mathcal{I}_0$ [AHV95].

Lemma 5.4. *The function* Φ_0 *is correct for* 0.

Proof. We prove by induction on the structure of $\psi \in \mathsf{Sub}(\phi)$ that $\mathsf{eval}^n(\Phi_0(\psi))$ is equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$ for all $n \geq 0$.

- If ψ is a CQ, then $\mathsf{eval}^n(\Phi_0(\psi)) = \mathsf{Ans}(\psi, \mathfrak{I}^{(0)}) = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$.
- If $\psi = \psi_1 \wedge \psi_2$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(\Phi_0(\psi_1)) \cap \mathsf{eval}^n(\Phi_0(\psi_2)) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0) \cap \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, 0) = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0),\end{aligned}$$

  and similarly for $\psi = \psi_1 \vee \psi_2$.
- If $\psi = \bigcirc^-\psi_1$, then $\mathsf{eval}^n(\Phi_0(\psi)) = \emptyset = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$.
- If $\psi = \bullet^-\psi_1$, then $\mathsf{eval}^n(\Phi_0(\psi)) = \Delta^{\mathsf{N_V}} = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$.
- If $\psi = \psi_1 \,\mathsf{S}\, \psi_2$, then Proposition 2.6 yields that

$$\mathsf{eval}^n(\Phi_0(\psi)) = \mathsf{eval}^n(\Phi_0(\psi_2)) = \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, 0) = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0).$$

- If $\psi = \bigcirc\psi_1$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(x_0^\psi)\\ &= \left\{\begin{array}{ll}\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0\\ \emptyset & \text{if } n = 0\end{array}\right\} = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0).\end{aligned}$$

- If $\psi = \bullet\psi_1$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(x_0^\psi)\\ &= \left\{\begin{array}{ll}\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0\\ \Delta^{\mathsf{N_V}} & \text{if } n = 0\end{array}\right\} = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0).\end{aligned}$$

- If $\psi = \psi_1 \,\mathsf{U}\, \psi_2$, then Proposition 2.6 yields that for $n > 0$ we have

$$\begin{aligned}\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(\Phi_0(\psi_2)) \cup \Big(\mathsf{eval}^n(\Phi_0(\psi_1)) \cap \mathsf{eval}^n(x_0^\psi)\Big)\\ &= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, 0) \cup \Big(\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0) \cap \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 1)\Big)\\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0),\end{aligned}$$

  and for $n = 0$ we get

$$\begin{aligned}\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(\Phi_0(\psi_2)) \cup \Big(\mathsf{eval}^n(\Phi_0(\psi_1)) \cap \mathsf{eval}^n(x_0^\psi)\Big)\\ &= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, 0) \cup \Big(\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0) \cap \emptyset\Big)\\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0).\end{aligned}$$

□

5.2 The Next Answer Formula

Assume now that $i > 0$ and we have computed a function $\Phi_{i-1} \colon \mathsf{Sub}(\phi) \to \mathsf{AF}_\phi^{i-1}$ that is correct for $i-1$ and contains only variables from Var_{i-1}^ϕ (this in particular holds for $i = 1$ as demonstrated in Section 5.1). We proceed as follows to construct a new function that is correct for i and contains only variables from Var_i^ϕ. We first define a function $\Phi_i^0 \colon \mathsf{Sub}(\phi) \to \mathsf{AF}_\phi^i$ similarly to Definition 5.3 that is correct for i, but may still contain variables with index $i-1$. We then iteratively replace these variables, starting with the ones for the smallest subformulae, until only variables with index i are left. In this process, we will ensure that correctness for i is preserved.

Definition 5.5. *Let $i > 0$ and $\Phi_{i-1} \colon \mathsf{Sub}(\phi) \to \mathsf{AF}_\phi^{i-1}$. We recursively define the mapping $\Phi_i^0 \colon \mathsf{Sub}(\phi) \to \mathsf{AF}_\phi^i$ as follows:*

- $\Phi_i^0(\psi_1) := \mathsf{Ans}(\psi_1, \mathfrak{I}^{(i)})$ *if ψ_1 is a CQ;*
- $\Phi_i^0(\psi_1 \wedge \psi_2) := \Phi_i^0(\psi_1) \cap \Phi_i^0(\psi_2)$; $\quad \Phi_i^0(\psi_1 \vee \psi_2) := \Phi_i^0(\psi_1) \cup \Phi_i^0(\psi_2)$

- $\Phi_i^0(\bigcirc\psi_1) := x_i^{\bigcirc\psi_1}$; $\quad \Phi_i^0(\bigcirc^-\psi_1) := \Phi_{i-1}(\psi_1)$;
- $\Phi_i^0(\bullet\psi_1) := x_i^{\bullet\psi_1}$; $\quad \Phi_i^0(\bullet^-\psi_1) := \Phi_{i-1}(\psi_1)$;
- $\Phi_i^0(\psi_1 \mathsf{U} \psi_2) := \Phi_i^0(\psi_2) \cup \Big(\Phi_i^0(\psi_1) \cap x_i^{\psi_1 \mathsf{U} \psi_2}\Big)$; *and*
- $\Phi_i^0(\psi_1 \mathsf{S} \psi_2) := \Phi_i^0(\psi_2) \cup \Big(\Phi_i^0(\psi_1) \cap \Phi_{i-1}(\psi_1 \mathsf{S} \psi_2)\Big)$.

The difference to Definition 5.3 is that the answer formulae for past operators are computed using the answer formulae for the previous time point.

Lemma 5.6. *If Φ_{i-1} is correct for $i-1$, then Φ_i^0 is correct for i.*

Proof. We prove by induction on the structure of $\psi \in \mathsf{Sub}(\phi)$ that $\mathsf{eval}^n(\Phi_i^0(\psi))$ is equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)$ for all $n \geq i$. The proof for most of the cases can easily be obtained from that of the corresponding cases in Lemma 5.4 by replacing 0 by i, 1 by $i+1$, and Φ_0 by Φ_i^0. We need to argue differently only for the past operators:

- If $\psi = \bigcirc^-\psi_1$ or $\psi = \bullet^-\psi_1$, then
$$\mathsf{eval}^n(\Phi_i^0(\psi)) = \mathsf{eval}^n(\Phi_{i-1}(\psi_1)) = \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i-1) = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)$$
since Φ_{i-1} is correct for $i-1 < i \leq n$.
- If $\psi = \psi_1 \mathsf{S} \psi_2$, then Proposition 2.6 yields that
$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_2)) \cup \Big(\mathsf{eval}^n(\Phi_i^0(\psi_1)) \cap \mathsf{eval}^n(\Phi_{i-1}(\psi))\Big) \\ &= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, i) \cup \Big(\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cap \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i-1)\Big) \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i).\end{aligned}$$
□

In order to remove the "old" variables with index $i-1$ from Φ_i^0, we can now substitute them by the values that we have just computed. 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 $\Phi_i^0(\psi)$. However, since this formula may itself contain another variable from $\mathsf{Var}_{i-1}^{\phi}$, we have to be careful about the order in which we do these substitutions. Since each $\Phi_i^0(\psi)$ 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. To formalize this intuition, we fix a total order $\psi^1 \prec \cdots \prec \psi^k$ on the set $\mathsf{FSub}(\phi) = \{\psi^1, \ldots, \psi^k\}$ such that whenever $\psi^j \in \mathsf{Sub}(\psi^{j'})$ for $j, j' \in \{1, \ldots, k\}$, then we have $j \leq j'$, i.e. $\psi^j = \psi^{j'}$ or $\psi^j \prec \psi^{j'}$. We can now define the substitutions for the variables $x_{i-1}^{\psi^1}, \ldots, x_{i-1}^{\psi^k}$ that constitute $\mathsf{Var}_{i-1}^{\phi}$.

Definition 5.7. *For $1 \leq j \leq k$ and $\psi \in \mathsf{Sub}(\phi)$, the answer formula $\Phi_i^j(\psi)$ is obtained by replacing every occurrence of $x_{i-1}^{\psi^j}$ in $\Phi_i^{j-1}(\psi)$ by*

$$\mathsf{update}\left(x_{i-1}^{\psi^j}\right) := \begin{cases} \Phi_i^{j-1}(\psi_1) & \text{if } \psi^j = \bigcirc\psi_1 \text{ or } \psi^j = \bullet\psi_1; \\ \Phi_i^{j-1}(\psi^j) & \text{if } \psi^j = \psi_1 \,\mathsf{U}\, \psi_2. \end{cases}$$

Finally, we set $\Phi_i := \Phi_i^k$.

It is easy to verify that each Φ_i^j as defined above is indeed a mapping from $\mathsf{Sub}(\phi)$ to AF_ϕ^i. The following lemma shows that these substitutions do not affect the correctness for i.

Lemma 5.8. *If Φ_i^{j-1} with $1 \leq j \leq k$ is correct for i, then Φ_i^j is also correct for i.*

Proof. Since eval^n is defined inductively on the structure of answer formulae, it suffices to show that $\mathsf{eval}^n(x_{i-1}^{\psi^j}) = \mathsf{eval}^n(\mathsf{update}(x_{i-1}^{\psi^j}))$ for all $n \geq i$. We make a case distinction on the form of ψ^j.

If $\psi^j = \bigcirc\psi_1$ or $\psi^j = \bullet\psi_1$, then we have $\mathsf{eval}^n(x_{i-1}^{\psi^j}) = \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i)$. Since Φ_i^{j-1} is correct for i, this is the same as $\mathsf{eval}^n(\Phi_i^{j-1}(\psi_1)) = \mathsf{eval}^n(\mathsf{update}(x_{i-1}^{\psi^j}))$.

If $\psi^j = \psi_1 \,\mathsf{U}\, \psi_2$, then we have $\mathsf{eval}^n(x_{i-1}^{\psi^j}) = \mathsf{Ans}(\psi^j, \mathfrak{I}^{(n)}, i)$. Again, since Φ_i^{j-1} is correct for i, this is the same as $\mathsf{eval}^n(\Phi_i^{j-1}(\psi^j)) = \mathsf{eval}^n(\mathsf{update}(x_{i-1}^{\psi^j}))$. □

5.3 The Algorithm

We are now ready to define our 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 $i \geq 0$. For this, consider the procedure described in Algorithm 1. This algorithm computes the functions Φ_i^j as described in the previous sections (see Figure 1).

Theorem 5.9. *Given a TCQ ϕ and an infinite sequence $\mathfrak{I} = (\mathcal{I}_i)_{i \geq 0}$ of interpretations, Algorithm 1 outputs $\mathsf{Ans}(\phi, \mathfrak{I}^{(i)})$ for each $i \geq 0$.*

Proof. For $i = 0$, the algorithm outputs $\mathsf{eval}^0(\Phi_0(\phi))$. Since Φ_0 is correct for 0 by Lemma 5.4, we have $\mathsf{eval}^0(\Phi_0(\phi)) = \mathsf{Ans}(\phi, \mathfrak{I}^{(0)})$ by Definition 5.2. For $i > 0$, it outputs $\mathsf{eval}^i(\Phi_i(\phi))$. Since Φ_{i-1} is correct for $i-1$, by Lemmas 5.6 and 5.8 we know that Φ_i is correct for i, and thus $\mathsf{eval}^i(\Phi_i(\phi)) = \mathsf{Ans}(\phi, \mathfrak{I}^{(i)})$. □

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, it is easy to see that our algorithm achieves a bounded history encoding and can be seen as a variant of the one from [Cho95] for less expressive queries.

Algorithm 1: The algorithm for computing the certain answers to a TCQ

Input : A TCQ ϕ and an infinite sequence $\mathfrak{I} = (\mathcal{I}_i)_{i \geq 0}$ of interpretations
Output : $\mathsf{Ans}(\phi, \mathfrak{I}^{(i)})$ for each $i \geq 0$

```
for i ← 0, 1, ... do
    if i = 0 then
        compute Φ_0;                          // according to Definition 5.3
    else
        compute Φ_i^0 from Φ_{i-1};           // according to Definition 5.5
        compute Φ_i^1, ..., Φ_i^k = Φ_i;      // according to Definition 5.7
    end
    output eval^i(Φ_i(φ));                    // according to Definition 5.2
end
```

If ϕ contains future operators, we still have to show that the space required to store (a representation of) Φ_i does not depend on i. For this, we first show that the image of Φ_i contains only variables from Var_i^ϕ, but no variables with smaller indices. This should be clear from the construction presented in the previous section, especially since we replace the variables x_{i-1}^ψ starting from those for the smallest subformulae. More formally, we show that all Φ_i are *monotone*, i.e. for all $\psi \in \mathsf{Sub}(\phi)$, the formula $\Phi_i(\psi)$ contains only variables from Var_i^ψ.

Lemma 5.10. *For all $i \geq 0$, the mapping Φ_i is monotone.*

Proof. The claim for $i = 0$ is easy to see from Definition 5.3. Assume now that Φ_{i-1} is monotone for some $i > 0$. We show that then Φ_i is also monotone by proving the following claim by induction on j, $0 \leq j \leq k$.

Claim. *For every $\psi \in \mathsf{Sub}(\phi)$, the answer formula $\Phi_i^j(\psi)$ contains only variables from Var_i^ψ and $\mathsf{Var}_{i-1}^\psi \cap \{x_{i-1}^{\psi^{j+1}}, \ldots, x_{i-1}^{\psi^k}\}$.*

For $j = 0$, we know from Definition 5.5 that for every $\psi \in \mathsf{Sub}(\phi)$ the answer formula $\Phi_i^0(\psi)$ contains only variables from Var_i^ψ and those occurring in $\Phi_{i-1}(\psi)$. Since Φ_{i-1} is monotone, it contains only variables from $\mathsf{Var}_{i-1}^\psi \subseteq \{x_{i-1}^{\psi^1}, \ldots, x_{i-1}^{\psi^k}\}$, and thus the claim is satisfied for $j = 0$.

Let now $0 < j \leq k$ and assume that the claim holds for $j - 1$. According to Definition 5.7, the answer formula Φ_i^j is obtained from Φ_i^{j-1} by replacing every occurrence of $x_{i-1}^{\psi^j}$ by $\mathsf{update}(x_{i-1}^{\psi^j})$. Since Φ_i^{j-1} satisfies the claim, it suffices to consider what happens to the variable $x_{i-1}^{\psi^j}$ in the image of Φ_i^{j-1}. By assumption, this variable can only occur in $\Phi_i^{j-1}(\psi)$ if $\psi^j \in \mathsf{FSub}(\psi)$. Thus, it is enough to show that $\mathsf{update}(x_{i-1}^{\psi^j})$ contains only variables from $\mathsf{Var}_i^{\psi^j}$. We prove this by a case distinction on the form of ψ^j.

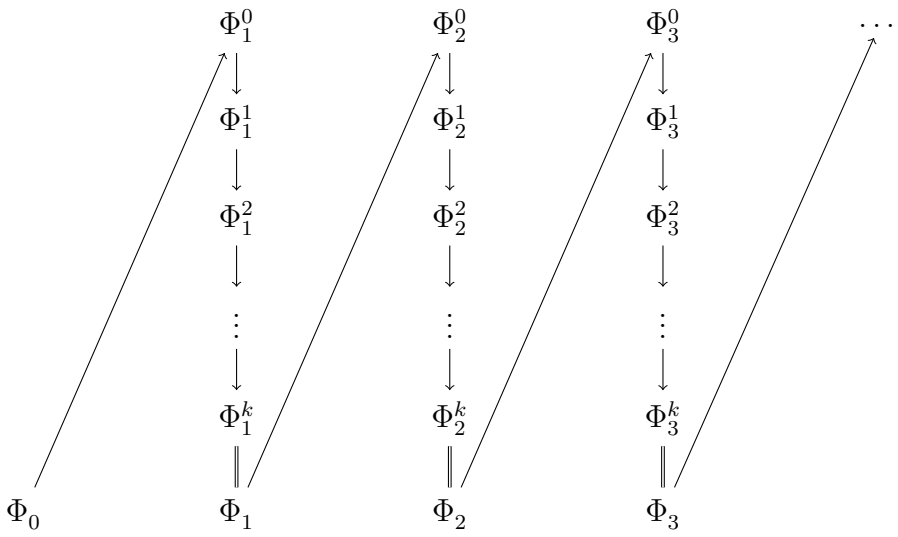

Figure 1: The order in which the mappings Φ_i^j are computed

- If $\psi^j = \bigcirc\psi_1$ or $\psi^j = \bullet\psi_1$, then $\mathsf{update}(x_{i-1}^{\psi^j}) = \Phi_i^{j-1}(\psi_1)$ by Definition 5.7. By the induction hypothesis, we have that $\Phi_i^{j-1}(\psi_1)$ contains only variables from $\mathsf{Var}_i^{\psi_1} = \mathsf{Var}_i^{\psi^j} \setminus \{x_i^{\psi^j}\}$ and $\mathsf{Var}_{i-1}^{\psi_1} \cap \{x_{i-1}^{\psi^j}, \ldots, x_{i-1}^{\psi^k}\}$. Note that the second set is empty since every variable $x_{i-1}^{\psi'} \in \mathsf{Var}_{i-1}^{\psi_1}$ must satisfy $\psi' \in \mathsf{FSub}(\psi_1) = \mathsf{FSub}(\psi^j) \setminus \{\psi^j\}$, and thus $\psi' \prec \psi^j$.

- If $\psi^j = \psi_1 \,\mathsf{U}\, \psi_2$, then $\mathsf{update}(x_{i-1}^{\psi^j}) = \Phi_i^{j-1}(\psi^j)$ by Definition 5.7. We have $\Phi_i^{j-1}(\psi^j) = \Phi_i^{j-1}(\psi_2) \cup (\Phi_i^{j-1}(\psi_1) \cap x_i^{\psi^j})$ by Definition 5.5 and the fact that Φ_i^{j-1} differs from Φ_i^0 only in the replacement of some of the variables with index $i-1$. By the induction hypothesis, each $\Phi_i^{j-1}(\psi_m)$, $m = 1, 2$, contains only variables from $\mathsf{Var}_i^{\psi_m} = \mathsf{Var}_i^{\psi^j} \setminus \{x_i^{\psi^j}\}$ and $\mathsf{Var}_{i-1}^{\psi_m} \cap \{x_{i-1}^{\psi^j}, \ldots, x_{i-1}^{\psi^k}\}$. By similar arguments as above, we can show that the second set is actually empty.

This finishes the proof of the claim and implies that for every $\psi \in \mathsf{Sub}(\phi)$, the answer formula $\Phi_i^k(\psi)$ contains only variables from Var_i^{ψ} and $\mathsf{Var}_{i-1}^{\psi} \cap \emptyset$, which concludes the proof of the lemma. $\square$

This shows that it is easy to compute the sets $\mathsf{eval}^i(\Phi_i(\phi)) = \mathsf{Ans}(\phi, \mathfrak{I}^{(i)})$ since each of the variables x_i^{ψ} simply has to be replaced by either $\emptyset$ or $\Delta^{\mathsf{N_V}}$ (see Definition 5.2). However, the variables in Var_i^{ϕ} may still occur several times in $\Phi_i(\phi)$. 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 $\Phi_1^0(\psi)$ (see

Definition 5.7), the variable x_1^ψ occurs twice in $\Phi_1^1(\phi) = \Phi_1(\phi)$. 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 (see Definition 5.2). Thus, the answer formula $\Phi_1(\phi)$ can be rewritten as follows:

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

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 $i > 0$. The expressions D_i can be fully evaluated by computing the specified unions and intersections of sets. Thus, the algorithm only has to store the two sets $A_i, D_i \subseteq \Delta^{\mathsf{Nv}}$ at each time point, i.e. we achieve a bounded history encoding, as in [Cho95].

The example demonstrates that it is important that the computed answer formulae are simplified at each step, while preserving their semantics under eval. We now show that a similar simplification can be achieved for every answer formula. For this, we first need to define what it means for two answer formulae to be equivalent.

Definition 5.11. *Two answer formulae $\alpha_1, \alpha_2 \in \mathsf{AF}_\phi^i$ are* equivalent *at i if for all $n \geq i$ it holds that $\mathsf{eval}^n(\alpha_1) = \mathsf{eval}^n(\alpha_2)$.*

Since i is usually clear from the context, we write $\alpha_1 \equiv \alpha_2$ if α_1 and α_2 are equivalent at i.

Lemma 5.12. *Every answer formula $\alpha \in \mathsf{AF}_\phi^i$ that contains only variables from Var_i^ϕ is equivalent to an answer formula of the form $\bigcup_{X \subseteq \mathsf{Var}_i^\phi} (A_X \cap \bigcap_{x \in X} x)$, where $A_X \subseteq \Delta^{\mathsf{Nv}}$ for each $X \subseteq \mathsf{Var}_i^\phi$.*[1]

Proof. We prove this by induction on the structure of α.

- If α is a set $A \subseteq \Delta^{\mathsf{Nv}}$, we define

$$A_X := \begin{cases} A & \text{if } X = \emptyset, \\ \emptyset & \text{otherwise.} \end{cases}$$

[1] Corresponding to eval, we consider the empty intersection to be Δ^{Nv}.

Hence we have

$$\begin{aligned}\mathsf{eval}^n\left(\bigcup_{X\subseteq \mathsf{Var}_i^\phi}\left(A_X\cap\bigcap_{x\in X}x\right)\right) &= (\mathsf{eval}^n(A)\cap\Delta^{\mathsf{N_V}})\cup\bigcup_{\substack{X\subseteq \mathsf{Var}_i^\phi\\ X\neq\emptyset}}\left(\emptyset\cap\bigcap_{x\in X}\mathsf{eval}^n(x)\right)\\ &= (\mathsf{eval}^n(A)\cap\Delta^{\mathsf{N_V}})\cup\emptyset\\ &= \mathsf{eval}^n(\alpha).\end{aligned}$$

- If α is a variable $x_i^\psi\in\mathsf{Var}_i^\phi$, let

$$A_X := \begin{cases}\Delta^{\mathsf{N_V}} & \text{if } X=\{x_j^\psi\},\\ \emptyset & \text{otherwise.}\end{cases}$$

Similar to the previous case, we obtain

$$\begin{aligned}\mathsf{eval}^n\left(\bigcup_{X\subseteq \mathsf{Var}_i^\phi}\left(A_X\cap\bigcap_{x\in X}x\right)\right) &= (\Delta^{\mathsf{N_V}}\cap\mathsf{eval}^n(x_i^\psi))\cup\emptyset\\ &= \mathsf{eval}^n(\alpha).\end{aligned}$$

- If $\alpha=\alpha_1\cup\alpha_2$, we have by the induction hypothesis:

$$\alpha_1\equiv\bigcup_{X\subseteq \mathsf{Var}_i^\phi}\left(B_X\cap\bigcap_{x\in X}x\right)\text{ and }\alpha_2\equiv\bigcup_{X\subseteq \mathsf{Var}_i^\phi}\left(C_X\cap\bigcap_{x\in X}x\right)$$

for some sets $B_X, C_X\subseteq\Delta^{\mathsf{N_V}}$ for each $X\subseteq\mathsf{Var}_i^\phi$. In this case, we can define $A_X := B_X\cup C_X$ to get

$$\begin{aligned}&\mathsf{eval}^n\left(\bigcup_{X\subseteq \mathsf{Var}_i^\phi}\left(A_X\cap\bigcap_{x\in X}x\right)\right)\\ &= \bigcup_{X\subseteq \mathsf{Var}_i^\phi}\left((B_X\cup C_X)\cap\bigcap_{x\in X}\mathsf{eval}^n(x)\right)\\ &= \bigcup_{X\subseteq \mathsf{Var}_i^\phi}\left(\mathsf{eval}^n\left(B_X\cap\bigcap_{x\in X}x\right)\cup\mathsf{eval}^n\left(C_X\cap\bigcap_{x\in X}x\right)\right)\\ &= \mathsf{eval}^n(\alpha_1)\cup\mathsf{eval}^n(\alpha_2)\\ &= \mathsf{eval}^n(\alpha).\end{aligned}$$

- If $\alpha=\alpha_1\cap\alpha_2$, we again have

$$\alpha_1\equiv\bigcup_{Y\subseteq \mathsf{Var}_i^\phi}\left(B_Y\cap\bigcap_{x\in Y}x\right)\text{ and }\alpha_2\equiv\bigcup_{Z\subseteq \mathsf{Var}_i^\phi}\left(C_Z\cap\bigcap_{x\in Z}x\right)$$

for suitable sets B_Y, C_Z. We now define

$$A_X := \bigcup_{\substack{Y,Z \subseteq \mathsf{Var}_i^\phi \\ Y \cup Z = X}} (B_Y \cap C_Z)$$

and obtain

$$\begin{aligned}
&\mathsf{eval}^n\left(\bigcup_{X \subseteq \mathsf{Var}_i^\phi}\left(A_X \cap \bigcap_{x \in X} x\right)\right) \\
&= \bigcup_{X \subseteq \mathsf{Var}_i^\phi}\left(\left(\bigcup_{\substack{Y,Z \subseteq \mathsf{Var}_i^\phi \\ Y \cup Z = X}} (B_Y \cap C_Z)\right) \cap \bigcap_{x \in X} \mathsf{eval}^n(x)\right) \\
&= \bigcup_{X \subseteq \mathsf{Var}_i^\phi} \bigcup_{\substack{Y,Z \subseteq \mathsf{Var}_i^\phi \\ Y \cup Z = X}}\left(B_Y \cap C_Z \cap \bigcap_{x \in X} \mathsf{eval}^n(x)\right) \\
&= \bigcup_{X \subseteq \mathsf{Var}_i^\phi} \bigcup_{\substack{Y,Z \subseteq \mathsf{Var}_i^\phi \\ Y \cup Z = X}}\left(\left(B_Y \cap \bigcap_{x \in Y} \mathsf{eval}^n(x)\right) \cap \left(C_Z \cap \bigcap_{x \in Z} \mathsf{eval}^n(x)\right)\right) \\
&= \bigcup_{Y \subseteq \mathsf{Var}_i^\phi} \bigcup_{Z \subseteq \mathsf{Var}_i^\phi}\left(\left(B_Y \cap \bigcap_{x \in Y} \mathsf{eval}^n(x)\right) \cap \left(C_Z \cap \bigcap_{x \in Z} \mathsf{eval}^n(x)\right)\right) \\
&= \bigcup_{Y \subseteq \mathsf{Var}_i^\phi} \mathsf{eval}^n\left(B_Y \cap \bigcap_{x \in Y} x\right) \cap \bigcup_{Z \subseteq \mathsf{Var}_i^\phi} \mathsf{eval}^n\left(C_Z \cap \bigcap_{x \in Z} x\right) \\
&= \mathsf{eval}^n(\alpha_1) \cap \mathsf{eval}^n(\alpha_2) \\
&= \mathsf{eval}^n(\alpha).
\end{aligned}$$

□

Consider now the answer formulae $\Phi_i(\psi)$ for $\psi \in \mathsf{Sub}(\phi)$. By Lemmata 5.4, 5.6, and 5.8, we have $\mathsf{eval}^n(\Phi_i(\psi)) = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)$ for every $n \geq i$. By Lemma 5.10, $\Phi_i(\psi)$ contains only variables from Var_i^ψ. Thus, by Lemma 5.12 there are sets $A_X \subseteq \Delta^{\mathsf{N_V}}$ for each $X \subseteq \mathsf{Var}_i^\psi$ such that $\Phi_i(\psi)$ is equivalent to

$$\alpha_i(\psi) := \bigcup_{X \subseteq \mathsf{Var}_i^\psi}\left(A_X \cap \bigcap_{x \in X} x\right)$$

at i. By Definition 5.11, this implies that

$$\mathsf{eval}^n(\alpha_i(\psi)) = \mathsf{eval}^n(\Phi_i(\psi)) = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)$$

holds for all $n \geq i$, i.e. $\alpha_i(\psi)$ is correct for i.

Thus, for the purposes of Algorithm 1, we can use $\alpha_i(\psi)$ in place of $\Phi_i(\psi)$, without changing its correctness. Observe that the size of $\alpha_i(\phi)$ is exponential in the size

of ϕ and $\mathsf{N_V}$, and polynomial in Δ, but independent of i. This shows that we have achieved a bounded history encoding, as in [Cho95].

In principle, our algorithm can be extended to deal with domain-independent queries with negation and first-order queries as atoms, as in [Cho95]. However, since we are interested in temporal OBDA over *DL-Lite*-TKBs and the reduction of Theorem 3.3 does not work with negation, we cannot allow negation in our query language.

6 Rigid Names

We now extend our temporal query language by designating certain concept names as being *rigid*, which means that their interpretation is not allowed to change over time. This especially makes sense regarding our application. For example, if the concept name Server describes the set of all severs, then it should be rigid since an application scenario with a server that stops being a server at some point in time would make no sense. The notion of rigidity has been explored for other temporal formalisms before [BGL12, BBL13].

For this purpose, we assume in this section that there is a set $\mathsf{N_{RC}} \subseteq \mathsf{N_C}$ of *rigid concept names*. In this setting, a finite sequence $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$ can only be a model of a TKB $\mathcal{K}$ if it fulfills the conditions of Definition 2.3 and additionally *respects* the rigid concept names, i.e. it satisfies $A^{\mathcal{I}_i} = A^{\mathcal{I}_j}$ for every rigid concept name A and all indices i, j between 0 and n.

For the remainder of this section, we restrict the query language to only allow so-called rooted CQs [Lut08]. Intuitively, these are CQs that refer to at least one named individual.

Definition 6.1. *A CQ ϕ is called* rooted *if*

(i) it contains at least one free variable or individual name, and

(ii) it is connected*, i.e. for all $x, y \in \mathsf{Var}(\phi) \cup \mathsf{Ind}(\phi)$ there is a sequence $x_1, \ldots, x_n \in \mathsf{Var}(\phi) \cup \mathsf{Ind}(\phi)$ such that $x_1 = x$, $x_n = y$, and for all i, $1 \leq i \leq n$, there is an $r \in \mathsf{N_R}$ such that either $r(x_i, x_{i+1}) \in \mathsf{At}(\phi)$ or $r(x_{i+1}, x_i) \in \mathsf{At}(\phi)$.*

This makes sense from an application point of view since one usually does not ask if there is some object with certain properties, but actually wants to know the names of all objects with these properties.

If we take the approach mentioned in Section 3.1 of viewing the input ABoxes as a temporal database and rewriting the TCQ into an ATSQL-query as in [CTB01], then the additional rigidity constraints can simply be enforced by triggers that

ensure that new knowledge about rigid names is added to the database at all previous time points.

However, the presence of rigid names poses a bigger problem for the incremental algorithms described in [Cho95] and Section 5. For example, if the ABox at the next time point includes the assertion $A(a)$, where A is rigid, then this retroactively also changes the answers to the query $A(x)$ at previous time points. However, the algorithms assume that the answers at previous time points do not change, and hence do not retain the whole history.

Before we consider how to modify the algorithms for temporal query answering over databases, we have to show that we can still employ the rewriting approach and answer atemporal queries over a TKB $\mathcal{K}$ by directly querying the database $\mathsf{DB}(\mathcal{K})$. This means that we have to reconsider the proof of Theorem 3.3 regarding the interpretation of rigid names. The main problem we have to solve is that the sequence $\mathfrak{I}_{\mathcal{K}}$ of canonical models does not necessarily respect the rigid concept names. In the following, let $\mathcal{K} = \langle (\mathcal{A}_i)_{i\geq 0}, \mathcal{T}\rangle$ be an infinite TKB. Similar to Section 5, we denote by $\mathcal{K}^{(n)} := \langle (\mathcal{A}_i)_{0\leq i\leq n}, \mathcal{T}\rangle$ the finite prefix of $\mathcal{K}$ of length $n+1$. We show how to construct modified sequences of interpretations (similar to $\mathfrak{I}_{\mathcal{K}^{(n)}}$ from Theorem 3.3) that respect rigid names.

The first step is to find a set $\mathcal{R} \subseteq \{A(a) \mid A \in \mathsf{N_{RC}}, a \in \mathsf{N_I}\}$ that specifies the rigid concept names that the individual names are allowed to satisfy. Of course, we have to ensure that the assertions in $\mathcal{R}$ are not contradicted by any of the ABoxes $\mathcal{A}_i$, $i \geq 0$. We construct $\mathcal{R}$ iteratively, starting from $\mathcal{R}_0 := \emptyset$, as follows. In each step, we add to $\mathcal{R}_j$, $j \geq 0$, all assertions $A(a)$ with $A \in \mathsf{N_{RC}}$ and $a \in \mathsf{N_I}$ that are implied by $\mathcal{A}_i \cup \mathcal{R}_j$ w.r.t. $\mathcal{T}$ for some $i \geq 0$, by which we mean that all models of $\mathcal{A}_i \cup \mathcal{R}_j$ and $\mathcal{T}$ are also models of $A(a)$. This reasoning task is called *instance checking* and can be done in polynomial time in *DL-Lite*$_{core}$ [CDL+09]. This results in a new set $\mathcal{R}_{j+1}$. We iterate this process until no new assertions are added. Since there are only polynomially many assertions of the form $A(a)$ as above, this is possible in polynomial time. We denote by $\mathcal{R}$ the final set computed by this procedure. We now show that, in order to answer TCQs over $\mathcal{K}^{(n)}$, we can equivalently consider the TKB $\mathcal{K}^{(n)}_{\mathcal{R}} := \langle (\mathcal{A}_i \cup \mathcal{R})_{0\leq i\leq n}, \mathcal{T}\rangle$.

Lemma 6.2. *Let ϕ be a TCQ and $\mathcal{K} = \langle (\mathcal{A}_i)_{i\geq 0}, \mathcal{T}\rangle$ be an infinite TKB. Then there is a set $\mathcal{R}$ as above such that, for all i and n with $0 \leq i \leq n$, we have*

$$\mathsf{Cert}(\phi, \mathcal{K}^{(n)}, i) = \mathsf{Cert}(\phi, \mathcal{K}^{(n)}_{\mathcal{R}}, i).$$

Proof. Let $\mathcal{R}$ be the set constructed above and $n \geq 0$. Obviously, every certain answer to ϕ w.r.t. $\mathcal{K}^{(n)}$ at some $i \geq 0$ is also a certain answer to ϕ w.r.t. $\mathcal{K}^{(n)}_{\mathcal{R}}$ at i. We show that all models of $\mathcal{K}^{(n)}$ must also satisfy $\mathcal{R}$ at each time point i, which proves the converse direction.

Let $\mathfrak{I} = (\mathcal{I}_i)_{0\leq i\leq n}$ be such that $\mathfrak{I} \models \mathcal{K}^{(n)}$ and assume that there is i, $0 \leq i \leq n$, with $\mathcal{I}_i \not\models \mathcal{R}$. Thus, there is $j > 0$ and $A(a) \in \mathcal{R}_j$ such that $\mathcal{I}_i \not\models A(a)$. Since

$\mathcal{I}_i \models \mathcal{A}_i$ and $\mathcal{I}_i \models \mathcal{T}$, by construction of $\mathcal{R}$ this means that there must be an assertion $B(b) \in \mathcal{R}_{j-1}$ such that $\mathcal{I}_i \not\models B(b)$. We can iterate this argument until we arrive at the fact that there is an assertion $C(c) \in \mathcal{R}_0$ such that $\mathcal{I}_i \not\models C(c)$. Since $\mathcal{R}_0 = \emptyset$, this is obviously a contradiction. □

Note that, if $\mathcal{R}$ is not consistent w.r.t. $\mathcal{T}$, this means that the TKB $\mathcal{K}$ is not consistent, i.e. there is no model of $\mathcal{K}$ that respects the rigid concept names.

Once we have computed $\mathcal{R}$, we can construct the desired sequence of canonical models that respects the rigid concept names. We start with the original sequence $\mathfrak{I}_{\mathcal{K}_{\mathcal{R}}^{(n)}} = (\mathcal{I}_{\mathcal{A}_i \cup \mathcal{R}, \mathcal{T}})_{0 \leq i \leq n}$ that was used in Theorem 3.3 (but now with $\mathcal{K}_{\mathcal{R}}^{(n)}$ instead of $\mathcal{K}^{(n)}$). It is important to note that these canonical models, as constructed in [CDL+09], are all countable. We define the set $\mathcal{D} \subseteq 2^{\mathsf{N_{RC}}}$ of subsets of $\mathsf{N_{RC}}$ that contains exactly the sets

$$\rho(\mathcal{I}_{\mathcal{A}_i \cup \mathcal{R}, \mathcal{T}}, x) := \{A \in \mathsf{N_{RC}} \mid x \in A^{\mathcal{I}_{\mathcal{A}_i \cup \mathcal{R}, \mathcal{T}}}\}$$

for all i, $0 \leq i \leq n$, and $x \in \Delta^{\mathcal{I}_{\mathcal{A}_i \cup \mathcal{R}, \mathcal{T}}}$. We will now modify each $\mathcal{I}_{\mathcal{A}_i \cup \mathcal{R}, \mathcal{T}}$ into a new interpretation $\mathcal{I}_i$ such that for each $Y \in \mathcal{D}$ there are countably infinitely many individuals $x \in \Delta^{\mathcal{I}_i}$ with $Y = \rho(\mathcal{I}_i, x)$.

To this end, consider i, n, $0 \leq i \leq n$, and $Y \in \mathcal{D}$. If $\mathcal{I}_{\mathcal{A}_i \cup \mathcal{R}, \mathcal{T}}$ does not contain any such individual, then we first have to add one. Fortunately, from the definition of $\mathcal{D}$ we know that there must be a j, $0 \leq j \leq n$, and $x \in \Delta^{\mathcal{I}_{\mathcal{A}_j \cup \mathcal{R}, \mathcal{T}}}$ such that $Y = \rho(\mathcal{I}_{\mathcal{A}_j \cup \mathcal{R}, \mathcal{T}}, x)$. To be on the safe side, we therefore construct the disjoint union $\mathcal{I}_i'$ of all interpretations in $\mathfrak{I}_{\mathcal{K}_{\mathcal{R}}^{(n)}}$, which means that we take the disjoint union of their domains, interpret the concept and role names by the union of the component interpretation of these names, and interpret the individual names as in $\mathcal{I}_{\mathcal{A}_i \cup \mathcal{R}, \mathcal{T}}$.

To ensure that there are even countably infinitely many such individuals, we make a similar construction as above by defining $\mathcal{I}_i''$ as the countably infinite disjoint union of $\mathcal{I}_i'$ with itself, where again the interpretation of the individual names remains unchanged. Finally, we ensure that all models have the same domain $\Delta := \mathsf{N_I} \cup (\mathcal{D} \times \mathbb{N})$ and interpret the individual names by the same domain elements by applying a simple bijection between the domain of each $\mathcal{I}_i''$ and Δ. In particular, each $a^{\mathcal{I}_i''}$ for $a \in \mathsf{N_I}$ is simply mapped to a, and every other element $x \in \Delta^{\mathcal{I}_i''}$ is mapped to some $(\rho(\mathcal{I}_i'', x), \ell)$ with $\ell \in \mathbb{N}$. We denote the resulting interpretation by $\mathcal{I}_i$ and define $\mathfrak{I}_{\mathcal{K}^{(n)}, \mathcal{R}} := (\mathcal{I}_i)_{0 \leq i \leq n}$.

Lemma 6.3. *The sequence $\mathfrak{I}_{\mathcal{K}^{(n)}, \mathcal{R}}$ is a model of $\mathcal{K}_{\mathcal{R}}^{(n)}$. Furthermore, for all rooted CQs ϕ and every i, $0 \leq i \leq n$, we have*

$$\mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}^{(n)}, \mathcal{R}}, i) = \mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}_{\mathcal{R}}^{(n)}}, i).$$

Proof. The interpretations $\mathcal{I}_i$ are still models of $\mathcal{T}$ since they are simply (renamed versions of) unions of models of $\mathcal{T}$. They are also still models of $\mathcal{A}_i \cup \mathcal{R}$ since

the interpretation of concept and role names on the individual names were never changed. Furthermore, the sequence $\mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}$ respects the rigid concept names since the elements of $\mathcal{D} \times \mathbb{N}$ always satisfy exactly the rigid names given by their first component, and every $a \in \mathsf{N_I}$ satisfies at least the rigid concept names A where $A(a) \in \mathcal{R}$. Assume now that we have $a^{\mathcal{I}_i} \in A^{\mathcal{I}_i}$ for some $A \in \mathsf{N_{RC}}$ and $a \in \mathsf{N_I}$, but $A(a) \notin \mathcal{R}$. By construction of $\mathcal{I}_i$, we thus also have $\mathcal{I}_{\mathcal{A}_i \cup \mathcal{R},\mathcal{T}} \models A(a)$. Since $A(a)$ is a CQ, by Proposition 3.2 all models of $\langle \mathcal{A}_i \cup \mathcal{R}, \mathcal{T} \rangle$ are also models of $A(a)$. But then we must have $A(a) \in \mathcal{R}$ by construction of $\mathcal{R}$, which contradicts the assumption that $A(a) \notin \mathcal{R}$. This shows that every $a \in \mathsf{N_I}$ satisfies exactly the rigid concept names A with $A(a) \in \mathcal{R}$ in each $\mathcal{I}_i$. Thus, $\mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}$ respects rigid names on the whole domain $\mathsf{N_I} \cup (\mathcal{D} \times \mathbb{N})$.

For the last part, we first show the inclusion from left to right. By definition of Ans it suffices to show that $\mathcal{I}_{\mathcal{A}_i \cup \mathcal{R},\mathcal{T}}$ is a model of a rooted *Boolean* CQ ϕ whenever $\mathcal{I}_i$ is a model of ϕ. Let π be a homomorphism of ϕ into $\mathcal{I}_i$. This can be used to define a homomorphism π' of ϕ into $\mathcal{I}_i''$ by simple composition with the bijection between $\Delta^{\mathcal{I}_i''}$ and $\mathsf{N_I} \cup (\mathcal{D} \times \mathbb{N})$. Similarly, we obtain a homomorphism π'' of ϕ into $\mathcal{I}_i'$ by taking for each $z \in \mathsf{Var}(\psi) \cup \mathsf{N_I}$ as $\pi''(z)$ the original element of $\Delta^{\mathcal{I}_i'}$ that gave rise to the copy $\pi'(z) \in \Delta^{\mathcal{I}_i''}$. Finally, since ϕ is rooted, the image of π'' must be contained in the original domain of $\mathcal{I}_{\mathcal{A}_i \cup \mathcal{R},\mathcal{T}}$,[2] and thus π'' is also a homomorphism of ϕ into $\mathcal{I}_{\mathcal{A}_i \cup \mathcal{R},\mathcal{T}}$. The converse inclusion can be shown by similar, but easier, arguments. □

We can now finally show the variant of Theorem 3.3 that can deal with rigid concept names.

Theorem 6.4. *Let ϕ be a rooted TCQ and $\mathcal{K} = \langle (\mathcal{A}_i)_{i \geq 0}, \mathcal{T} \rangle$ be an infinite TKB. Then there is a set $\mathcal{R}$ as above such that, for all i and n with $0 \leq i \leq n$, we have*

$$\mathsf{Cert}(\phi, \mathcal{K}_{\mathcal{R}}^{(n)}, i) = \mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}, i) = \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K}_{\mathcal{R}}^{(n)}), i).$$

Proof. Let $\mathcal{R}$ be the set constructed above. The inclusion $\mathsf{Cert}(\phi, \mathcal{K}_{\mathcal{R}}^{(n)}, i) \subseteq \mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}, i)$ can be shown as in the proof of Theorem 3.3 since we know by Lemma 6.3 that $\mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}$ is a model of $\mathcal{K}_{\mathcal{R}}^{(n)}$. The inclusion $\mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K}_{\mathcal{R}}^{(n)}), i) \subseteq \mathsf{Cert}(\phi, \mathcal{K}_{\mathcal{R}}^{(n)}, i)$ directly follows.

It remains to show the inclusion $\mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}, i) \subseteq \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K}_{\mathcal{R}}^{(n)}), i)$, for which we again employ induction on the structure of ϕ. The only difference to the corresponding induction proof for Theorem 3.3 is the base case where ϕ is a CQ; all the other cases can be shown as before. But for every rooted CQ ϕ, Lemma 6.3 and Proposition 3.2 imply that $\mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}, i) = \mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}_{\mathcal{R}}^{(n)}}, i) = \mathsf{Ans}(\phi, \mathcal{I}_{\mathcal{A}_i \cup \mathcal{R},\mathcal{T}}) = \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{A}_i \cup \mathcal{R})) = \mathsf{Ans}(\phi^{\mathcal{T}}, \mathsf{DB}(\mathcal{K}_{\mathcal{R}}^{(n)}), i)$. □

[2] In the disjoint union, there are no role connections between the components.

Algorithm 2: The algorithm for computing the certain answers to a rooted TCQ in the presence of rigid names

Input : A rooted TCQ ϕ and an infinite TKB $\mathcal{K} = \langle (\mathcal{A}_i)_{i \geq 0}, \mathcal{T} \rangle$
Output : $\mathsf{Cert}(\phi, \mathcal{K}^{(i)})$ for each $i \geq 0$

```
Active ← ∅
for R ∈ ℜ do
    if R is consistent w.r.t. T then
        initialize an instance A_R of Algorithm 1 on input φ^T
        Active ← Active ∪ {R}
    end
end
for i ← 0, 1, ... do
    for R ∈ Active do
        if A_i ∪ R is consistent w.r.t. T then
            run A_R on input DB(A_i ∪ R) to compute Cert(φ, K_R^(i))
        else
            terminate A_R
            Active ← Active \ {R}
        end
    end
    output ⋂_{R∈Active} Cert(φ, K_R^(i))
end
```

Note that $\mathsf{DB}(\mathcal{K}_{\mathcal{R}}^{(n)})$ is independent of the construction of $\mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}$, and we can now simply apply the algorithm of Section 5 to the modified sequence of interpretations $\mathsf{DB}(\mathcal{K}_{\mathcal{R}}^{(n)})$ instead of $\mathsf{DB}(\mathcal{K}^{(n)})$. More formally, let $\mathfrak{R}$ denote the set of all sets $\mathcal{R}$ of the form described above. Algorithm 2 describes the steps necessary to compute the certain answers to a TCQ in the presence of rigid names.

For each $\mathcal{R} \in \mathfrak{R}$ that is consistent w.r.t. $\mathcal{T}$, we start an instance $\mathsf{A}_{\mathcal{R}}$ of Algorithm 1. All these instances are run in parallel, with the only difference between them being that each instance has a fixed set $\mathcal{R}$ of assumptions about the rigid names. If we discover one of these assumptions to be inconsistent w.r.t. one of the ABoxes $\mathcal{A}_i$, the corresponding instance is stopped. All remaining instances $\mathsf{A}_{\mathcal{R}}$ compute the certain answers to ϕ relative to $\mathcal{R}$, and the actual set of certain answers to ϕ is computed by taking the intersection over all these sets.

Theorem 6.5. *Given a rooted TCQ ϕ and an infinite TKB $\mathcal{K} = \langle (\mathcal{A}_i)_{i \geq 0}, \mathcal{T} \rangle$, Algorithm 2 outputs $\mathsf{Cert}(\phi, \mathcal{K}^{(i)})$ for each $i \geq 0$.*

Proof. Observe first that from a theoretical point of view, the consistency tests in Algorithm 2 are not necessary since if $\mathcal{R}$ or any $\mathcal{A}_i \cup \mathcal{R}$ is inconsistent w.r.t. $\mathcal{T}$,

then $\mathcal{K}_{\mathcal{R}}^{(i)}$ has no models, and thus $\mathsf{Cert}(\phi, \mathcal{K}_{\mathcal{R}}^{(i)}) = \Delta^{\mathsf{N_V}}$ does not contribute to the computation of the intersection in Algorithm 2.[3] We thus consider here the variant of Algorithm 2 without consistency tests, which means that Active is always equal to $\mathfrak{R}$.

By Theorems 5.9 and 6.4, we know that the instances $\mathsf{A}_{\mathcal{R}}$ for $\mathcal{R} \in \mathfrak{R}$ compute the set $\mathsf{Cert}(\phi, \mathcal{K}_{\mathcal{R}}^{(i)})$ at each time point $i \geq 0$. In particular, this set always contains the set $\mathsf{Cert}(\phi, \mathcal{K}^{(i)})$. Furthermore, by Lemma 6.2, we know that there must be at least one $\mathcal{R} \in \mathfrak{R}$ such that $\mathsf{Cert}(\phi, \mathcal{K}_{\mathcal{R}}^{(i)})$ is even equal to $\mathsf{Cert}(\phi, \mathcal{K}^{(i)})$. By construction, this particular $\mathcal{R}$ passes all consistency tests of Algorithm 2 (if $\mathcal{K}$ is consistent at all). This shows that the intersection at the end of each loop of Algorithm 2 yields the set $\mathsf{Cert}(\phi, \mathcal{K}^{(i)})$. □

We have thus extended Algorithm 1—and by extension also the algorithm described in [Cho95]—to deal with rigid concept names in rooted TCQs.

7 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 [CDL+09]. 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 [CTB01].

The second approach works by eliminating the future operators and evaluating the resulting query using the algorithm of [Cho95], 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. Then, we presented an algorithm that works directly with the future operators. We showed that the algorithm computes exactly the desired answers and also achieves a bounded history encoding.

Finally, we also described an approach to extend the proposed algorithm to deal with rigid concept names if only rooted CQs are allowed. In future work, we will investigate how to adapt the algorithm to deal also with rigid role names and compare the performance of all three described approaches on temporal databases.

[3] We again consider answers to be mappings $\mathfrak{a}\colon \mathsf{N_V} \to \Delta$ (here we have $\Delta = \mathsf{N_I}$) instead of $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \Delta$ to simplify the presentation.

References

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[AKRZ09] Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. *DL-Lite* with temporalised concepts, rigid axioms and roles. In Silvio Ghilardi and Roberto Sebastiani, editors, *Proc. of the 6th Int. Symp. on Frontiers of Combining Systems (FroCoS'09)*, volume 5749 of *Lecture Notes in Computer Science*, pages 133–148. Springer-Verlag, 2009.

[AKRZ10] Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. Temporal conceptual modelling with DL-Lite. In *Proc. of the 2010 Int. Workshop on Description Logics (DL'10)*, volume 573 of *CEUR-WS*, 2010.

[AKRZ12] Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. A cookbook for temporal conceptual data modelling with description logics. *CoRR*, abs/1209.5571, 2012.

[BBL13] Franz Baader, Stefan Borgwardt, and Marcel Lippmann. Temporalizing ontology-based data access. In Maria Paola Bonacina, editor, *Proc. of the 24th Int. Conf. on Automated Deduction (CADE'13)*, volume 7898 of *Lecture Notes in Artificial Intelligence*, pages 330–344. Springer-Verlag, 2013.

[BGL12] Franz Baader, Silvio Ghilardi, and Carsten Lutz. LTL over description logic axioms. *ACM Transactions on Computational Logic*, 13(3):21:1–21:32, 2012.

[CDL+09] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, and Riccardo Rosati. Ontologies and databases: The DL-Lite approach. In Sergio Tessaris, Enrico Franconi, Thomas Eiter, Claudio Gutierrez, Siegfried Handschuh, Marie-Christine Rousset, and Renate A. Schmidt, editors, *Reasoning Web, 5th Int. Summer School 2009, Tutorial Lectures*, volume 5689 of *Lecture Notes in Computer Science*, pages 255–356. Springer-Verlag, 2009.

[CGL+05] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI'05)*, pages 602–607. AAAI Press, 2005.

[Cho95] Jan Chomicki. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Transactions on Database Systems*, 20(2):148–186, 1995.

[CM77] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison, editors, *Proc. of the 9th Annual ACM Symp. on Theory of Computing (STOC'77)*, pages 77–90. ACM Press, 1977.

[CTB01] Jan Chomicki, David Toman, and Michael H. Böhlen. Querying AT-SQL databases with temporal logic. *ACM Transactions on Database Systems*, 26(2):145–178, 2001.

[Gab89] Dov Gabbay. Declarative past and imperative future. In Behnam Banieqbal, Howard Barringer, and Amir Pnueli, editors, *Proc. of the 1987 Coll. on Temporal Logic in Specification*, volume 398 of *Lecture Notes in Computer Science*, pages 409–448. Springer-Verlag, 1989.

[GK12] Víctor Gutiérrez-Basulto and Szymon Klarman. Towards a unifying approach to representing and querying temporal data in description logics. In Markus Krötzsch and Umberto Straccia, editors, *Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems (RR'12)*, volume 7497 of *Lecture Notes in Computer Science*, pages 90–105. Springer-Verlag, 2012.

[HS91] Klaus Hülsmann and Gunter Saake. Theoretical foundations of handling large substitution sets in temporal integrity monitoring. *Acta Informatica*, 28(4):365–407, 1991.

[LMS02] François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. Temporal logic with forgettable past. In *Proc. of the 17th Annual IEEE Symp. on Logic in Computer Science (LICS'02)*, pages 383–392. IEEE Press, 2002.

[Lut08] Carsten Lutz. The complexity of conjunctive query answering in expressive description logics. In *Proc. of the 4th Int. Joint Conf. on Automated Reasoning (IJCAR'08)*, volume 5195 of *Lecture Notes in Artificial Intelligence*, pages 179–193. Springer-Verlag, 2008.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Proc. of the 18th Annual Symp. on Foundations of Computer Science (SFCS'77)*, pages 46–57, 1977.

[SL89] Gunter Saake and Udo W. Lipeck. Using finite-linear temporal logic for specifying database dynamics. In Egon Börger, Hans Kleine Büning, and Michael M. Richter, editors, *Proc. of the 2nd Workshop on*

Computer Science Logic (CSL'88), volume 385 of *Lecture Notes in Computer Science*, pages 288–300. Springer-Verlag, 1989.

[Tom04] David Toman. Logical data expiration. In Jan Chomicki, Ron van der Meyden, and Gunter Saake, editors, *Logics for Emerging Applications of Databases*, chapter 6, pages 203–238. Springer-Verlag, 2004.

[Wil99] Thomas Wilke. Classifying discrete temporal properties. In *Proc. of the 16th Annual Symp. on Theoretical Aspects of Computer Science (STACS'99)*, volume 1563 of *Lecture Notes in Computer Science*, pages 32–46. Springer-Verlag, 1999.