Temporalizing Rewritable Query Languages over Knowledge Bases

Stefan Borgwardt, Marcel Lippmann, Veronika Thost

Institute of Theoretical Computer Science, Technische Universität Dresden, 01062 Dresden, Germany

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. OBDA can sometimes be realized by compiling the information of the ontology into the query and the database. The resulting query is then answered using classical database techniques.

In this paper, we consider a temporal version of OBDA. We propose a generic temporal query language that combines linear temporal logic with queries over ontologies. This language is well-suited for expressing temporal properties of dynamic systems and is useful in context-aware applications that need to detect specific situations. We show that, if atemporal queries are rewritable in the sense described above, then the corresponding temporal queries are also rewritable such that we can answer them over a temporal database. We present three approaches to answering the resulting queries.

Keywords: Ontology-Based Data Access, Linear Temporal Logic, Query Answering, Rewritability, Description Logic

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 [1]. However, we cannot assume that the sensors provide a complete description of the current state of the environment. Thus, the closed world assumption employed by database systems (i.e., facts not present in the database are assumed to be false) is not appropriate since there may be facts of which the truth is not known. For example, a sensor for specific 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, some knowledge about its behavior is often available. This background knowledge could be used to formulate constraints on the behavior of the real environment. These constraints help formulate queries to detect more complex situations.

This information (i.e., the sensor data and the background knowledge) is stored in so-called *knowledge bases*, which are sometimes called *ontologies*. A knowledge base consists of a fact base and a theory, which store the data in a formally well-understood way. The *fact base* contains simple facts (e.g., the concrete values given by sensors), and is interpreted with the *open world assumption*, i.e., facts not present are assumed to be unknown rather than false. The *theory* contains the additional background knowledge (e.g., general domain knowledge) stored in a symbolic representation. The situations to be detected are then specified in an appropriate query language. The resulting queries are then evaluated w.r.t. the information encoded in the knowledge base. This general approach is often called ontology-based data access (OBDA) [2, 3].

However, since the environment is changing, it is often desirable to specify situations that take into account *temporal* behavior. In this setting, we model the incoming information as a *sequence* of fact bases, one for each moment in time in which the system has been observed. To recognize situations, we propose to add a temporal logical component to atemporal queries over knowledge bases. We use the operators of the temporal logic LTL, which allows to reason about a linear and discrete flow of time [4]. 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 requires 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 distributed video platform providing several services such as uploading, streaming, and

Email addresses: stefborg@tcs.inf.tu-dresden.de (Stefan Borgwardt), lippmann@tcs.inf.tu-dresden.de (Marcel Lippmann), thost@tcs.inf.tu-dresden.de (Veronika Thost) transcoding (i.e., the conversion of video formats). At any given time point, a fact base for such a system could contain facts like the following, which describe that there is a server s with an overutilized CPU c, which executes an uploading service (ULS) p_1 and a transcoding service (TCS) p_2, both of which are active:

$$\begin{array}{llll} \mathsf{CPU}(c), & \mathsf{Overutilized}(c), & \mathsf{Server}(s), & \mathsf{hasCPU}(s,c), \\ \mathsf{ULS}(p_1), & \mathsf{executes}(s,p_1), & \mathsf{Active}(p_1), & \\ \mathsf{TCS}(p_2), & \mathsf{executes}(s,p_2), & \mathsf{Active}(p_2) & \end{array}$$

The background theory could contain an axiom such as

$$\forall x.\mathsf{Server}(x) \wedge \big(\exists y.\mathsf{hasCPU}(x,y) \wedge \mathsf{Overutilized}(y)\big) \\ \rightarrow \mathsf{Overloaded}(x),$$

which states that a server having an overutilized CPU is overloaded. Given the above fact base, we can conclude that s is currently overloaded.

Since transcoding is very resource-intensive, it is important to transcode popular videos preemptively in phases of less utilization instead of on demand in phases of high utilization. However, the situation can clearly change after a preemptive transcoding service has been started. For that reason, one may want to detect critical situations in which a server of the platform has become overloaded while executing such a service.

The temporal query

$$\mathsf{TCS}(x) \wedge \mathsf{Server}(y) \wedge \mathsf{executes}(y,x) \wedge \psi_0 \wedge \big(\mathsf{NLB}(y)\,\mathsf{S}\,\bigcirc\psi_t\big)$$

with

$$\psi_t := \begin{cases} \mathsf{Active}(x) \wedge \mathsf{Overloaded}(y) & \text{if } t = 0 \\ \psi_0 \wedge \bigcirc\Diamond\psi_{t-1} & \text{if } t \geq 1 \end{cases}$$

and $t \geq 0$ therefore asks for a transcoding service x and a server y that executes it, where x is active and y is overloaded. The second part of the query requires that $\mathsf{NLB}(y)$ has been true for the whole time *since* (S) the subquery $\bigcirc\psi_t$ was true. In other words, we are looking for a time point in the past that satisfies ψ_t such that all time points since then satisfy $\mathsf{NLB}(y)$, which expresses that y has not been affected by a load balancing operation in the meantime. The subquery ψ_t again asks for x to be active and y to be overloaded, and furthermore that there is a time point after the current one ($\bigcirc\Diamond$) satisfying ψ_{t-1}. We are thus asking for a series of $t+1$ critical time points (not necessarily immediately following each other). We consider the temporal behavior of this example query in more detail in Sections 5 and 6.

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 like S or $\Box^-$. We also show that in our setting it is indeed always possible to construct an equivalent query using only past operators (see Section 5.3). 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 formula results in a blowup that is at least exponential and no constructions of size less than triply exponential are known [5].

1.1. Related Work

In this paper, we consider so-called *rewritable query languages*, i.e., query languages for which evaluating a query over a knowledge base can be reduced to answering a rewritten query (in a different language) over a database induced by the knowledge base. Such query languages, especially in the context of Description Logics (DLs) [6], are covered extensively in the literature (see Example 2.11).

Investigations of *temporal* query languages based on combinations of query languages and temporal logics such as LTL [4] have started only quite recently. Yet, a number of very expressive temporal query languages have been proposed [7–10].

For rewritable query languages, most research focuses on light-weight languages of the *DL-Lite* family [11]. However, instead of temporalizing the query language and evaluating the queries over a global knowledge base, also temporal knowledge bases are examined, which allow temporal operators to occur inside axioms. These approaches are based on research about temporalized description logics (see [12] for a survey). For example, in [13], various light-weight DLs are extended by allowing the temporal operators to interfere with the DL component. Following the ideas of [13], in [14] a rewritable temporal query language over temporal knowledge bases in *DL-Lite* is proposed.

There is also a lot of closely related work in the field of temporal databases. In [15], for instance, the authors describe a temporal extension of the SQL query language that can answer temporal queries over a temporal database. In [16–18], an approach is described that reduces the amount of space needed to evaluate temporal queries by keeping only the relevant data in the database instead of keeping track of all the information from the past.

1.2. Our Contribution

In this paper, we consider temporal queries over knowledge bases in a very general setting that allows us to extend many existing atemporal query languages by temporal operators (cf. Section 3). In Section 4, we show that the reasoning task of *temporal OBDA* in this setting can be reduced to answering queries over temporal databases. The main part of the paper is thus concerned with what we call the *temporal database monitoring problem*, where a fixed temporal query is continuously evaluated over a temporal sequence of databases.

We present three approaches to solving this problem. The first one employs existing temporal database systems using a translation from our temporal query language into a specialized database query language [15] (cf. Section 5.1). The second approach again rewrites the query in order to obtain a query without future operators, which then can be answered using an algorithm from [16] (cf. Section 5.3). The advantage of this algorithm is that the time required to answer the temporal query at the current time point does not depend on the total running time of the system; this is called a *bounded history encoding* in [16]. In Section 6, we propose a new algorithm that extends the one from [16] in that it also deals with future operators directly while guaranteeing a bounded history encoding. We also discuss different advantages and drawbacks of the three approaches.

Sometimes it is desired to state that certain facts do not change over time, i.e., are *rigid*. In Section 7, we show how our proposed algorithm can be extended to deal with a limited form of rigidity in a specific class of queries.

This paper is an extension of [19], where we have considered only the special case of answering temporal queries over *DL-Lite*$_{core}$-ontologies. In contrast to [19], we also show in this paper that our proposed algorithm preserves the bounded history encoding of [16]. Additionally, this paper contains the full proofs of our results. To improve readability, some of them are presented in the appendix.

2. Preliminaries

As mentioned in the introduction, we consider temporal queries over knowledge bases in a very general setting. This section describes the logical framework for querying atemporal knowledge bases and basic properties of this framework we require for the rest of the paper. We also give a wealth of examples of concrete query formalisms from the literature that satisfy our restrictions.

2.1. Logics

Our basic setting is that of function-free first-order languages. In any such language, we need to assert the truth of ground facts.

Definition 2.1 (assertion). Let $\mathsf{N_C}$ be a set of *constants*, and let $(\mathsf{N_P^n})_{n\geq 0}$ be a family of sets of *n-ary predicate symbols.* An *assertion* is an expression of the form $P(c_1,\dots,c_n)$ for $P\in\mathsf{N_P^n}$ and $c_1,\dots,c_n\in\mathsf{N_C}$.

An *interpretation* is a pair $\mathcal{I}=(\Delta^\mathcal{I},\cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty set (called the *domain* of $\mathcal{I}$) and $\cdot^\mathcal{I}$ is an *interpretation function* that assigns to every $P\in\mathsf{N_P^n}$ an n-ary relation $P^\mathcal{I}\subseteq\Delta^n$, and to every $c\in\mathsf{N_C}$ an element $c^\mathcal{I}\in\Delta^\mathcal{I}$. Such an interpretation is called *finite* if its domain is finite. Two interpretations are *isomorphic* if there is a bijective mapping between their domains that preserves the interpretations of all constants and predicate symbols.

We say that $\mathcal{I}$ is a *model* of an assertion $P(c_1,\dots,c_n)$, written $\mathcal{I}\models P(c_1,\dots,c_n)$, if $(c_1^\mathcal{I},\dots,c_n^\mathcal{I})\in P^\mathcal{I}$.

To simplify the presentation of our results, we assume in the following that the sets $\mathsf{N_C}$ and $\bigcup_{n\geq 0}\mathsf{N_P^n}$ are non-empty and finite, i.e., we restrict to finitely many symbols that are relevant for some domain of interest. We further assume that the sets of constants and predicate symbols are disjoint.

By using *axioms* that are more expressive than simple assertions, more elaborate properties of interpretations can be stated. In a logical formalism, *theories* are usually finite sets of axioms. In the following, we consider a generic *logic*, which consists of a set of theories expressible in it, together with a satisfaction relation.

Definition 2.2 (logic). A *logic* is a pair $(\mathcal{L},\models_\mathcal{L})$, where $\mathcal{L}$ is a set of *$\mathcal{L}$-theories* and $\models_\mathcal{L}$ is a *satisfaction relation* between interpretations and $\mathcal{L}$-theories, i.e., $\models_\mathcal{L}\subseteq\mathbb{I}\times\mathcal{L}$, where $\mathbb{I}$ denotes the set of all interpretations. For an interpretation $\mathcal{I}$ and an $\mathcal{L}$-theory $\mathcal{T}$, we write $\mathcal{I}\models_\mathcal{L}\mathcal{T}$ if $(\mathcal{I},\mathcal{T})\in\models_\mathcal{L}$. In this case, we also say that $\mathcal{I}$ is a *model* of $\mathcal{T}$.

In many concrete logics, there is a basic satisfaction relation for axioms that is lifted in a natural way to theories. However, some logics put further restrictions on the shape of their theories apart from them being a set of axioms. This is the reason why we choose to define logics as sets of theories rather than sets of axioms.

In the following, we often refer to a logic by its first component $\mathcal{L}$, which is implicitly associated with an entailment relation $\models_\mathcal{L}$. If the logic is clear from the context, we may also write $\models$ instead of $\models_\mathcal{L}$, and simply speak of *theories.*

Definition 2.3 (knowledge base). Given a logic $\mathcal{L}$, a *knowledge base* over $\mathcal{L}$ is a pair $\mathcal{K}=\langle\mathcal{A},\mathcal{T}\rangle$, where $\mathcal{A}$ is a finite set of assertions, called *fact base*, and $\mathcal{T}$ is an $\mathcal{L}$-theory.

We write $\mathcal{I}\models\mathcal{A}$, and say that $\mathcal{I}$ is a *model* of $\mathcal{A}$, if $\mathcal{I}$ is a model of all assertions in the fact base $\mathcal{A}$. A knowledge base $\mathcal{T}=\langle\mathcal{A},\mathcal{T}\rangle$ is *consistent* if there is an interpretation that is a model of both $\mathcal{A}$ and $\mathcal{T}$.

A basic requirement for the logics considered in this paper is that consistency should be decidable. The consistency check is the first step of any reasoning algorithm, as an inconsistent knowledge base makes most reasoning problems trivial.

Example 2.4. The main instances of our framework we will describe in more detail are based on *Description Logics (DLs)* [6]. In these formalisms, the language is restricted to unary and binary predicates, called *concept names* and *role names*, respectively. So-called *concept constructors* are used to build more expressive unary predicates, called *concepts*, from these basic names. Similarly, more complex *roles*, i.e., binary predicates, can be built.

In this setting, theories are made up from axioms like *general concept inclusions (GCIs)* of the form $C\sqsubseteq D$, which restrict all models to interpret C by a subset of the interpretation of D, and similar axioms between roles. Sometimes additional conditions are imposed on the left-hand side or the right-hand side of such inclusions. In DLs, such theories are usually called *TBoxes* or *ontologies.*

Often, the axioms of a DL are expressible as sentences of first-order logic.

The expressivity of DLs ranges from light-weight DLs such as members of the *DL-Lite* family [20] and $\mathcal{EL}$ [21] to the very expressive $\mathcal{SROIQ}$, which forms the basis for the standardized Semantic Web ontology language OWL 2 [22]. However, a major criterion in their design is that consistency of knowledge bases should be decidable.

For the purposes of this paper, we are particularly interested in so-called *Horn* description logics. They are distinguished by an inability to express disjunction, which leads to the interesting property that knowledge bases can often be characterized in terms of a single *canonical model* (see Definition 2.8). To this family belong many members of the *DL-Lite* family, extensions of $\mathcal{EL}$, and syntactically restricted forms of more expressive DLs like Horn-$\mathcal{SHIQ}$ [23, 24].

A different logical formalism is *Datalog* [1], which is based on rules of the form $Q \leftarrow P_1 \wedge \cdots \wedge P_m$, where each atom is of the form $P(z_1, \ldots, z_n)$ for $P \in \mathsf{N}_\mathsf{P}^n$ and variables or constants z_i, with the restriction that every variable that occurs in the *head* Q must also occur in the *body* $P_1 \wedge \cdots \wedge P_m$. Thus, rules without body are simply assertions. Theories are finite sets of such rules and are called *Datalog programs*. For the satisfaction relation, the usual first-order reading of the rules is employed, where all variables are universally quantified. An interesting property of Datalog is that every program $\mathcal{P}$ has a *least Herbrand model*, which contains exactly those assertions that hold in all models of $\mathcal{P}$ (similar to the canonical models of knowledge bases in Horn-DLs). Since we do not consider function symbols, the Herbrand domain is N_C, and thus this least Herbrand model is finite. There is also *linear Datalog*, where the body of any rule may contain at most one atom that also occurs in the head of a rule. Theories of logics in the *Datalog*$^\pm$ family [25] consist of *tuple-generating dependencies* that generalize Datalog rules in that they allow new (existentially quantified) variables to occur in the head.

2.2. Queries

We stay just as generic in the description of query languages over $\mathcal{L}$.

Definition 2.5 (query language). Let N_V be a set of *variables*, disjoint from N_C and N_P^n. A *variable assignment* is a mapping of the form $\mathfrak{a}\colon \{x_1, \ldots, x_n\} \to \mathsf{N}_\mathsf{C}$ with $x_1, \ldots, x_n \in \mathsf{N}_\mathsf{V}$.

A *query language* is a triple $(\mathcal{Q}, \mathsf{FVar}, \models_\mathcal{Q})$, where $\mathcal{Q}$ is a set of $\mathcal{Q}$-*queries*, $\mathsf{FVar}\colon \mathcal{Q} \to 2^{\mathsf{N}_\mathsf{V}}$ maps every $\mathcal{Q}$-query to the finite set of its *free variables*, and $\models_\mathcal{Q}$ is a *satisfaction relation*, denoted as $\mathcal{I} \models_\mathcal{Q} \mathfrak{a}(\psi)$ for an interpretation $\mathcal{I}$, a $\mathcal{Q}$-query ψ, and a variable assignment $\mathfrak{a}\colon \mathsf{FVar}(\psi) \to \mathsf{N}_\mathsf{C}$,[1] such that

(i) for all $\psi \in \mathcal{Q}$, $\mathfrak{a}_1, \mathfrak{a}_2\colon \{x_1, \ldots, x_n\} \to \mathsf{N}_\mathsf{C}$, and interpretations $\mathcal{I}$ with $\mathfrak{a}_1(x_i)^\mathcal{I} = \mathfrak{a}_2(x_i)^\mathcal{I}$, $1 \le i \le n$, we have $\mathcal{I} \models_\mathcal{Q} \mathfrak{a}_1(\psi)$ iff $\mathcal{I} \models_\mathcal{Q} \mathfrak{a}_2(\psi)$; and

(ii) for all $\psi \in \mathcal{Q}$, $\mathfrak{a}\colon \{x_1, \ldots, x_n\} \to \mathsf{N}_\mathsf{C}$, and isomorphic interpretations $\mathcal{I}_1, \mathcal{I}_2$, we have $\mathcal{I}_1 \models_\mathcal{Q} \mathfrak{a}(\psi)$ iff $\mathcal{I}_2 \models_\mathcal{Q} \mathfrak{a}(\psi)$.

If $\mathcal{I} \models_\mathcal{Q} \mathfrak{a}(\psi)$, we say that $\mathfrak{a}$ is an *answer* to ψ w.r.t. $\mathcal{I}$.

Conditions (i) and (ii) above are reasonable assumptions for query languages that express that satisfaction does not depend on the names of domain elements, only on their interpretation. We include them in this definition since they are needed in the proof of Theorems 4.1 and 7.5 to unify the domains of several interpretations, and at the end of Section 4 to simplify the presentation of the temporal database monitoring problem.

We adopt the same conventions as for logics and, e.g., refer to query languages only by their first component and write $\models$ if $\mathcal{Q}$ is clear from the context. We further denote by $\mathsf{Ans}(\psi, \mathcal{I}) \subseteq \mathsf{N}_\mathsf{C}^{\mathsf{FVar}(\psi)}$ the set of all answers to a query ψ w.r.t. an interpretation $\mathcal{I}$. For convenience, if there is an implicit total order $x_1 < \cdots < x_n$ on the elements of $\mathsf{FVar}(\psi) = \{x_1, \ldots, x_n\}$, we sometimes denote variable assignments $\mathfrak{a}\colon \{x_1, \ldots, x_n\} \to \mathsf{N}_\mathsf{C}$ by tuples of the form $(\mathfrak{a}(x_1), \ldots, \mathfrak{a}(x_n))$.

We now lift the semantics of queries to deal with knowledge bases. The main notion is that of *certain answers* to a query, which are variable assignments that satisfy the query in all models of a given knowledge base.

Definition 2.6 (certain answer). Let $\mathcal{L}$ be a logic, $\mathcal{Q}$ a query language, $\mathcal{K}$ a knowledge base, and ψ a query. A variable assignment $\mathfrak{a}\colon \mathsf{FVar}(\psi) \to \mathsf{N}_\mathsf{C}$ is called a *certain answer* to ψ w.r.t. $\mathcal{K}$, written $\mathcal{K} \models \mathfrak{a}(\psi)$, if for every model $\mathcal{I}$ of $\mathcal{K}$, it holds that $\mathfrak{a}$ is an answer to ψ w.r.t. $\mathcal{I}$.

Similar to before, we denote by $\mathsf{Cert}(\psi, \mathcal{K}) \subseteq \mathsf{N}_\mathsf{C}^{\mathsf{FVar}(\psi)}$ the set of all certain answers to a query ψ w.r.t. a knowledge base $\mathcal{K}$. The problem of computing $\mathsf{Cert}(\psi, \mathcal{K})$ from ψ and $\mathcal{K}$ is called *query answering*.

A special situation arises when the considered queries have no free variables. Queries of this form are called *Boolean* queries since the set $\mathsf{Cert}(\psi, \mathcal{K})$ can only be empty or contain the empty variable assignment as its only element. In the latter case, we say that ψ is *entailed* by $\mathcal{K}$, and write $\mathcal{K} \models \psi$, if $\mathsf{Cert}(\psi, \mathcal{K})$ is non-empty. Similarly, we write $\mathcal{I} \models \psi$ for an interpretation $\mathcal{I}$ if $\mathsf{Ans}(\psi, \mathcal{I})$ is non-empty.

Example 2.7. The simplest query language arises from considering all assertions as Boolean queries, and taking $\models_\mathcal{Q}$ to be $\models$ (ignoring the variable assignments). The entailment of an assertion by a knowledge base is then equivalent to the usual definition.

Similarly, we can consider the Boolean query language $\mathcal{Q} := \mathcal{L}$ with $\models_\mathcal{Q}$ given by $\models_\mathcal{L}$, i.e., we can ask for the

[1]We do not consider variable assignments that do not map exactly the free variables of the query.

entailment of theories. In the context of Description Logics, an important such query language is that of *subsumptions* which ask for the entailment of single GCIs $C \sqsubseteq D$, i.e., whether the concept C is a subconcept of D in all models of a given knowledge base.

One step up from assertion queries are so-called *instance queries (IQs)* of the form $P(z_1, \ldots, z_n)$, where $P \in \mathsf{N}_\mathsf{P}^n$ and each z_i may be either a constant or a variable. The free variables of this query are simply the variables among $z_1, \ldots, z_n$. To compute $\mathsf{Cert}(\psi, \mathcal{K})$, we have to determine all variable assignments that *certainly* (in all models of $\mathcal{K}$) make the assertion true when replacing the free variables accordingly.

For relational databases, an important class of queries are *conjunctive queries (CQs)* (also called *select-project-join queries*) of the form $\exists y_1, \ldots, y_m.\psi$, where $y_1, \ldots, y_m \in \mathsf{N}_\mathsf{V}$ and ψ is a conjunction of instance queries [1]. As usual, the free variables of this CQ are those occurring in it, except $y_1, \ldots, y_m$. In contrast to the free variables, which range only over the constants, the quantified variables $y_1, \ldots, y_m$ range over the whole domain of a given interpretation. The semantics of CQs is thus obtained by viewing them as first-order sentences in the obvious way.

In the database setting, one is concerned with computing $\mathsf{Ans}(\psi, \mathcal{I})$ for a conjunctive query ψ and a *finite* interpretation $\mathcal{I}$, which can be seen as a relational database. This can be done by asking, e.g., an SQL query over this database. The more general problem of computing certain answers to conjunctive queries w.r.t. a knowledge base has been investigated for many logical formalisms, in particular DLs [26–29]. To solve it, sometimes the so-called *first-order-rewritability* of CQs w.r.t. the logic $\mathcal{L}$ is exploited (see Definition 2.10).

In this approach, so-called *first-order queries* are used to capture the answers of a CQ w.r.t. a knowledge base. These queries allow arbitrary nesting of all usual constructs of first-order logic, including negation and universal quantification. The essential part of the reduction is that these first-order queries only have to be answered over *finite* interpretations, i.e., databases. In this setting, first-order-rewritability is actually equivalent to rewritability into much simpler *unions* (disjunctions) of conjunctive queries (UCQs) [30]. Another class of interest between UCQs and arbitrary first-order queries are *positive existential queries (PEQs)* of the form $\exists y_1, \ldots, y_m.\psi$, where ψ is a positive Boolean combination of instance queries (i.e., using conjunction and disjunction, but no negation).

In the context of Description Logics, where the predicates are restricted to be at most binary, *conjunctive regular path queries (CRPQs)* generalize conjunctive queries in a different direction by allowing conjuncts of the form $L(x, y)$, where L is a regular expression over the binary predicate symbols [31, 32]. In an interpretation over this signature, which is essentially a labeled graph, these conjuncts express the existence of a path from x to y such that the concatenation of its edge labels belongs to the language generated by L.

We will also consider *Datalog queries* $(\mathcal{P}, P)$, where $\mathcal{P}$ is a Datalog program and P is the *goal predicate* to be answered [1]. The free variables are $x_1, \ldots, x_n$, where n is the arity of P. The program $\mathcal{P}$ uses auxiliary predicates that are local to the query and used to evaluate it. Only auxiliary predicates are allowed to occur in the heads of rules, and the goal predicate P must be an auxiliary predicate. A variable assignment $\mathfrak{a}$ is an answer to such a query w.r.t. an interpretation $\mathcal{I}$ if all extensions of $\mathcal{I}$ to the auxiliary predicates that satisfy $\mathcal{P}$ also satisfy $P(\mathfrak{a}(x_1), \ldots, \mathfrak{a}(x_n))$. This is equivalent to the containment of this assertion in the least Herbrand model of $\langle \mathsf{facts}(\mathcal{I}), \mathcal{P} \rangle$, where $\mathsf{facts}(\mathcal{I})$ denotes the (finite) set of all assertions that $\mathcal{I}$ is a model of.

In particular, every UCQ can be formulated as a Datalog query in which the goal predicate is the only auxiliary predicate, which furthermore does not occur in the body of any rule. Similarly, PEQs correspond to Datalog queries with *nonrecursive* programs [1].

In this paper, we assume that every query language contains a special Boolean query true, which holds in all interpretations. Likewise, we assume the presence of a Boolean query false that does not hold in any interpretation. It is straightforward to add these to a query language without affecting any of the properties or constructions described in the following.

2.3. Canonical Models and Rewritability

We now come to the first important restriction that we make on the logics and query languages we consider.

Definition 2.8 (canonical model). A logic $\mathcal{L}$ has the *canonical model property* w.r.t. a query language $\mathcal{Q}$ if every consistent knowledge base $\mathcal{K}$ has a countably infinite *canonical model* $\mathcal{I}_\mathcal{K}$, which is a model of $\mathcal{K}$ with the property that for all queries ψ, we have $\mathsf{Cert}(\psi, \mathcal{K}) = \mathsf{Ans}(\psi, \mathcal{I}_\mathcal{K})$.

Canonical models are sometimes called *universal models.*

The restriction to countably infinite canonical models is a technical one, which ensures that all these models have the same cardinality. This is not a great restriction since canonical models are often explicitly constructed in a countable way. However, if the canonical model is finite, one can usually add countably infinitely many copies of it without changing the answers. We exploit this to unify the domains of different interpretations for Theorems 4.1 and 7.5.

Example 2.9. The following table lists several DLs $\mathcal{L}$ and query languages $\mathcal{Q}$ that have the canonical model property. The canonical model is usually obtained by applying the axioms of the knowledge base $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ as *completion rules* to the facts in $\mathcal{A}$ in order to obtain a model of $\mathcal{K}$ (this is also called *chase* in database theory).

In the case of [33, 34], it is constructed from the least Herbrand model of a Datalog program that depends on $\mathcal{A}$ and $\mathcal{T}$.

The result from [34] also holds for Horn-$\mathcal{SHIQ}$ w.r.t. CQs that use only simple roles (i.e., roles without transitive subroles).

$\mathcal{L}$	$\mathcal{Q}$	shown in
$\mathcal{EL}^{++}$	subs.	[35]
$\mathit{DL\text{-}Lite}_{\mathcal{R}/\mathcal{F}}$	UCQ	[26, Theorem 29]
$\mathcal{ELH}$	UCQ	[36, Lemma 1]
$\mathcal{ELI}^f$	CQ	[37, Lemma 5]
$\mathcal{ELH}_{\bot}^{dr}$	CQ	[28, Proposition 4]
$\mathit{DL\text{-}Lite}_{horn}^{\mathcal{N}}$	CQ	[38, Theorem 4]
$\mathit{DL\text{-}Lite}_{horn}$	PEQ	[39, Theorem 3]
$\mathcal{ELHI}^{\neg}$	CQ	[33, Lemma 10]
Horn-$\mathcal{ALCHIQ}$	CQ	[34, Theorem 3], [24]
Horn-$\mathcal{ALCHOIQ}_{\mathsf{Self}}^{\mathsf{Disj}}$	CRPQ	[40, Theorem 2]

For computing the set of certain answers to a query, an important approach is to rewrite the query such that it can be evaluated over a single *finite* interpretation, i.e., a database. Generally, the interpretation and the rewritten query together contain the information of the theory and the original query, whereas the knowledge from the fact base only influences the definition of the interpretation.

This is called the *combined approach* to rewriting [38, 39], in contrast to the original idea [20, 26], where the finite interpretation is obtained by simply viewing the fact base under the closed world assumption. There, all necessary information of the theory and the original query is encoded in the rewritten query. With both approaches, the rewritten query usually belongs to a more expressive query language.

Definition 2.10 (rewritable). Let $\mathcal{L}$ be a logic and $\mathcal{Q}_1$, $\mathcal{Q}_2$ be query languages. We say that $\mathcal{Q}_1$-queries are $\mathcal{Q}_2$-*rewritable* w.r.t. $\mathcal{L}$ if one can compute

- for every theory $\mathcal{T}$, a finite set $\Delta_{\mathcal{T}}$ that contains $\mathsf{N_C}$,
- for every consistent knowledge base $\mathcal{K}$, a finite interpretation $\mathcal{D}_{\mathcal{K}}$ over the domain $\Delta_{\mathcal{T}}$ such that $c^{\mathcal{D}_{\mathcal{K}}} = c$ holds for all $c \in \mathsf{N_C}$, and
- for every $\mathcal{Q}_1$-query ψ and theory $\mathcal{T}$, a $\mathcal{Q}_2$-query $\psi^{\mathcal{T}}$ such that $\mathsf{FVar}(\psi) = \mathsf{FVar}(\psi^{\mathcal{T}})$,

such that for all *consistent* knowledge bases $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ and $\mathcal{Q}_1$-queries ψ, we have $\mathsf{Cert}(\psi, \mathcal{K}) = \mathsf{Ans}(\psi^{\mathcal{T}}, \mathcal{D}_{\mathcal{K}})$.

To summarize, $\mathcal{Q}_2$-rewritability means that finding certain answers to $\mathcal{Q}_1$-queries w.r.t. $\mathcal{L}$ can be reduced to finding (ordinary) answers to $\mathcal{Q}_2$-queries over finite interpretations, which can be seen as relational databases. This brings us to our last requirement, namely that the set of answers to a $\mathcal{Q}_2$-query w.r.t. a finite interpretation should be computable. In case of $\mathcal{Q}_2$-rewritability of $\mathcal{Q}_1$-queries w.r.t. $\mathcal{L}$, this implies that the set of answers to a $\mathcal{Q}_1$-query w.r.t. a knowledge base is also computable.

In [20], where *first-order-rewritability* was introduced for conjunctive queries in *DL-Lite*, the rewritten first-order query $\psi^{\mathcal{T}}$ was called the *perfect reformulation* of ψ (w.r.t. $\mathcal{T}$). The term *perfect* refers to the fact that this query can then be used to answer the original query over any fact base. Recall that first-order-rewritability is equivalent to UCQ-rewritability, but first-order queries can be more concise than UCQs.

The above definition is an extension of this original version of rewritability that captures more results that have been shown since then. It contains some technical restrictions that are needed to lift this to the temporal setting (see Theorem 4.1), but which are satisfied by all instances described in Example 2.11 below. Most importantly, the construction of $\mathcal{D}_{\mathcal{K}}$ is independent of a concrete query, and likewise, $\psi^{\mathcal{T}}$ does not depend on a fact base.

It is clear that finiteness of $\mathcal{D}_{\mathcal{K}}$ is not sufficient in practice, where one would additionally like to have small interpretations $\mathcal{D}_{\mathcal{K}}$ over which $\mathcal{Q}_2$-queries can be evaluated efficiently. Indeed, many rewritability results have subsequently been refined to improve this behavior. However, we are not so much interested in the theoretical complexity of answering queries as our approach to temporal queries will anyway always need to compute the whole set $\mathsf{Ans}(\psi^{\mathcal{T}}, \mathcal{D}_{\mathcal{K}})$, which is already exponential in the cardinality of $\mathsf{FVar}(\psi)$. For details, see the discussion after Lemma 6.10.

Example 2.11. Below, we list several rewritability results for different instances of $\mathcal{L}$, $\mathcal{Q}_1$, and $\mathcal{Q}_2$, where $\mathrm{FO}_=$ denotes first-order queries with equality and UCQ^+ a combination of a UCQ with a linear Datalog program.

For the logics of the *DL-Lite* and $\mathcal{EL}$ families, the finite interpretation $\mathcal{D}_{\mathcal{K}}$ is usually obtained by viewing the fact base under the closed world assumption, but sometimes additional constant symbols are introduced. In the other cases, $\mathcal{D}_{\mathcal{K}}$ is based on the least Herbrand model of a suitable Datalog program constructed from $\mathcal{K}$.

The result of [41] applies only to so-called rooted acyclic CQs; however, the rewriting is more efficient than that of [26] when measured in combined complexity.

Again, the result from [34] also holds for Horn-$\mathcal{SHIQ}$ if the CQs do not contain non-simple roles.

The constructions for $\mathcal{LDL}^+$ and $\mathcal{SROEL}(\sqcap, \times)$ do not rewrite the query, and therefore these logics also have the canonical model property.

To ensure termination of the rewriting algorithm in [42], the theories have to be restricted, e.g., to *linear* or *sticky* sets of tuple-generating dependencies.

$\mathcal{L}$	$\mathcal{Q}_1$	$\mathcal{Q}_2$	shown in
$\mathcal{EL}^{++}$	subs.	subs.	[35]
DL-Lite$_{\mathcal{R}}$	CQ	UCQ	[26, Lemma 39]
$\mathcal{ELH}^{dr}_{\bot}$	CQ	FO$_=$	[28, Theorem 5]
DL-Lite$^{\mathcal{N}}_{horn}$	CQ	FO$_=$	[38, Theorem 10]
DL-Lite$_{\mathcal{R}}$	UCQ	PEQ	[43, Theorem 2]
DL-Lite	CQ	UCQ	[41, Theorem 5]
$\mathcal{ELHI}^{\neg}$	CQ	Datalog	[33, Theorem 2 and Lemma 16]
DL-Lite$_{\mathcal{R}}$	CQ	UCQ	[33, Theorem 2 and Lemma 16]
DL-Lite^{+}	CQ	UCQ^{+}	[33, Theorem 2 and Lemma 16]
Horn-$\mathcal{ALCHIQ}$	CQ	UCQ	[34, Theorem 4]
$\mathcal{LDL}^{+}$	IQ	IQ	[44, Corollary 11]
$\mathcal{SROEL}(\sqcap, \times)$	IQ	IQ	[45, 46]
Datalog$^{\pm}$ family	CQ	UCQ	[42, Theorem 1]

It was suggested in [30, 47] that one should consider rewritability as a decision problem, and ask, for a given logic $\mathcal{L}$ and a $\mathcal{Q}_1$-query, whether it is $\mathcal{Q}_2$-rewritable. In case of decidability, one can consider instead of $\mathcal{Q}_1$ only those elements of $\mathcal{Q}_1$ that have this property, and thus obtain another instance of Definition 2.10.

3. Temporal Queries

In the following, let $\mathcal{L}$ be a logic and $\mathcal{Q}$ a query language. We now lift the definitions of the previous section to a temporal setting, where we have a global theory describing the background knowledge of a domain and a sequence of fact bases that represent preprocessed sensor data obtained at successive points in time.

Definition 3.1 (temporal knowledge base). Given a logic $\mathcal{L}$, a *temporal knowledge base* (TKB) over $\mathcal{L}$ is a pair $\mathcal{K} = \langle(\mathcal{A}_i)_{0\leq i\leq n}, \mathcal{T}\rangle$ consisting of a finite sequence of fact bases $\mathcal{A}_i$ and an $\mathcal{L}$-theory $\mathcal{T}$.

Let $\mathfrak{I} = (\mathcal{I}_i)_{0\leq i\leq n}$ be a finite sequence of interpretations $\mathcal{I}_i = (\Delta, \cdot^{\mathcal{I}_i})$ over a fixed non-empty domain Δ. Then, $\mathfrak{I}$ is a *model* of $\mathcal{K}$ (written $\mathfrak{I} \models \mathcal{K}$) if $\mathcal{I}_i \models \mathcal{A}_i$ and $\mathcal{I}_i \models \mathcal{T}$ for all i, $0 \leq i \leq n$. A TKB is *consistent* if it has a model.

We consider only sequences of interpretations that satisfy the *constant domain assumption*, i.e., they are defined over a common domain. Thus, we assume that the world does not change, only the predicates defined in it may evolve.

Although similar to what was done in [9, 10], our temporal query language can in principle be based on any atemporal query language $\mathcal{Q}$. Another difference to those approaches is that we do not allow negation as this would destroy the rewritability properties of $\mathcal{Q}$ (see Theorem 4.1).

Definition 3.2 (temporal query). Given a query language $\mathcal{Q}$, *temporal $\mathcal{Q}$-queries* are built from $\mathcal{Q}$-queries as follows:

- every $\mathcal{Q}$-query ψ is a temporal $\mathcal{Q}$-query; and
- if ϕ_1 and ϕ_2 are temporal $\mathcal{Q}$-queries, 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), $\Diamond^-\phi_1$ (some time in the past),
  - $\phi_1 \,\mathsf{U}\, \phi_2$ (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*.

As usual, if $\mathcal{Q}$ is clear from the context, we use the term *temporal queries (TQs)*. The set $\mathsf{FVar}(\phi)$ of *free variables* of a TQ ϕ is defined as the union of the sets $\mathsf{FVar}(\psi)$ of all queries ψ occurring in ϕ. A TQ ϕ is called *Boolean* if $\mathsf{FVar}(\phi) = \emptyset$. We further denote by $\mathsf{Sub}(\phi)$ the set of all TQs occurring as temporal subqueries in ϕ (including ϕ itself). For a subquery ϕ_1 of ϕ, we denote by $\mathfrak{a}_{\phi_1}$ the restriction of a variable assignment $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \mathsf{N_C}$ to $\mathsf{FVar}(\phi_1)$.

Definition 3.3 (semantics of TQs). Let ϕ be a TQ, $\mathfrak{I} = (\mathcal{I}_i)_{0\leq i\leq n}$ a sequence of interpretations over a common domain, $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \mathsf{N_C}$ a variable assignment, and i be an integer with $0 \leq i \leq n$. The *satisfaction relation* $\mathfrak{I}, i \models \mathfrak{a}(\phi)$ is defined by induction on the structure of ϕ as follows:

ϕ	$\mathfrak{I}, i \models \mathfrak{a}(\phi)$ iff
$\mathcal{Q}$-query ψ	$\mathcal{I}_i \models \mathfrak{a}(\psi)$
$\phi_1 \wedge \phi_2$	$\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1)$ and $\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2)$
$\phi_1 \vee \phi_2$	$\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1)$ or $\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2)$
$\bigcirc\phi_1$	$i < n$ and $\mathfrak{I}, i+1 \models \mathfrak{a}(\phi_1)$
$\bullet\phi_1$	$i < n$ implies $\mathfrak{I}, i+1 \models \mathfrak{a}(\phi_1)$
$\bigcirc^-\phi_1$	$i > 0$ and $\mathfrak{I}, i-1 \models \mathfrak{a}(\phi_1)$
$\bullet^-\phi_1$	$i > 0$ implies $\mathfrak{I}, i-1 \models \mathfrak{a}(\phi_1)$
$\Box\phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all k, $i \leq k \leq n$
$\Box^-\phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all k, $0 \leq k \leq i$
$\Diamond\phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for some k, $i \leq k \leq n$
$\Diamond^-\phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for some k, $0 \leq k \leq i$
$\phi_1 \,\mathsf{U}\, \phi_2$	there is k, $i \leq k \leq n$, with $\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2)$ and $\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1)$ for all j, $i \leq j < k$
$\phi_1 \,\mathsf{S}\, \phi_2$	there is k, $0 \leq k \leq i$, with $\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2)$ and $\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1)$ for all j, $k < j \leq i$

If $\mathfrak{I}, i \models \mathfrak{a}(\phi)$, then $\mathfrak{a}$ is called an *answer* to ϕ w.r.t. $\mathfrak{I}$ at time point i. Given a TKB $\mathcal{K} = \langle(\mathcal{A}_i)_{0\leq i\leq n}, \mathcal{T}\rangle$, we say that $\mathfrak{a}$ is a *certain answer* to ϕ w.r.t. $\mathcal{K}$ at time point i, written $\mathcal{K}, i \models \mathfrak{a}(\phi)$, if for all models $\mathfrak{I}$ of $\mathcal{K}$, 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 $\mathsf{Cert}(\phi, \mathcal{K}) := \mathsf{Cert}(\phi, \mathcal{K}, n)$. A Boolean TQ ϕ is *entailed* by $\mathcal{K}$ (at time point i) if the set $\mathsf{Cert}(\phi, \mathcal{K})$ ($\mathsf{Cert}(\phi, \mathcal{K}, i)$) is non-empty. In this case, we write $\mathcal{K} \models \phi$ ($\mathcal{K}, i \models \phi$), and similarly for $\mathfrak{I} \models \phi$ and $\mathfrak{I}, i \models \phi$.

Here we assume that there is no time point before 0 or after n, similar to the temporal semantics used for LTL in [48] or for temporal query languages for databases [16, 49, 50]. This semantics has the effect that the temporal query $\bigcirc\mathsf{true}$ is not entailed at the last time point. This may seem counterintuitive, but it makes sense in our scenario since we do not know whether the system we observe is still running at the next point in time.

Alternatively, we could adopt the more common semantics based on infinite sequences of interpretations, the first n of which must be models of the respective fact bases. However, this in turn has some unintended consequences. Since we want to monitor systems based on the available facts, it is natural to restrict the aggregation operators to the time points for which sensor data is available. For example, if we ask for all processes that have always been running using the query $\mathsf{Process}(x) \wedge \Box^-\mathsf{Running}(x)$, then time points before the system was started ($i < 0$) are not relevant. Likewise, we may want to ask about a property that always held from a specific time point *up to now*, regardless of what happens in the future.

A compromise between our semantics and one based on infinite sequences of interpretations could be obtained by "looping" the last interpretation or fact base infinitely often, which means that the facts of the last time point stay valid forever. This would make $\bigcirc\mathsf{true}$ equivalent to true, while retaining the spirit of the finite semantics. However, this semantics also has counterintuitive side-effects as it makes severe assumptions on the future behavior of the observed system.

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 thus, at the first time point, $\phi_1 \,\mathsf{S}\, \phi_2$ is equivalent to ϕ_2 since $\bigcirc^-(\phi_1 \,\mathsf{S}\, \phi_2)$ does not have any answers.

Proposition 3.4. *For* $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \mathsf{N_C}$ *and* $0 < i \leq n$, *we have* $\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \,\mathsf{S}\, \phi_2)$ *iff*

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

Furthermore, $\mathfrak{I}, 0 \models \mathfrak{a}(\phi_1 \,\mathsf{S}\, \phi_2)$ *iff* $\mathfrak{I}, 0 \models \mathfrak{a}_{\phi_2}(\phi_2)$.

Similar equivalences hold for U, $\Diamond$, and $\Diamond^-$. To be able to employ analogous reductions for $\Box$ and $\Box^-$, we use the operators $\bullet$ and $\bullet^-$ that are tautological at the last and first time point, respectively.

4. Rewriting Temporal Queries

To answer temporal queries, we lift the rewriting approach introduced in Section 2.3 to the temporal setting. We recall the basic assumptions we made on the query languages $\mathcal{Q}_1, \mathcal{Q}_2$ and the logic $\mathcal{L}$:

- Consistency of knowledge bases in $\mathcal{L}$ should be decidable. This is a basic prerequisite for any reasoning procedure, in particular for query answering.
- The logic $\mathcal{L}$ should have the canonical model property w.r.t. $\mathcal{Q}_1$ (see Definition 2.8). This property is often a first step towards a rewritability result. For our temporal setting, it is an important ingredient to the proof of Theorem 4.1 below.
- $\mathcal{Q}_1$-queries should be $\mathcal{Q}_2$-rewritable w.r.t. $\mathcal{L}$. In particular, we will make heavy use of the objects $\Delta_{\mathcal{T}}$, $\mathcal{D}_{\mathcal{K}}$, and $\psi^{\mathcal{T}}$ introduced in Definition 2.10.
- Last but not least, the set of answers to any $\mathcal{Q}_2$-query w.r.t. a finite interpretation should be computable.

Under all of these assumptions, we can show that temporal $\mathcal{Q}_1$-queries enjoy a similar rewritability property w.r.t. knowledge bases formulated in $\mathcal{L}$, and thus we can compute the certain answers to temporal $\mathcal{Q}_1$-queries over $\mathcal{L}$.

We first lift the constructions of Definitions 2.8 and 2.10 to the temporal setting. For this, consider a temporal $\mathcal{Q}_1$-query ϕ and a consistent TKB $\mathcal{K} = \langle(\mathcal{A}_i)_{0\leq i\leq n}, \mathcal{T}\rangle$. Obviously, the atemporal knowledge bases $\mathcal{K}_i := \langle\mathcal{A}_i, \mathcal{T}\rangle$, $0 \leq i \leq n$, are then also consistent, and thus we can define the sequences $\mathfrak{I}_{\mathcal{K}} := (\mathcal{I}_{\mathcal{K}_i})_{0\leq i\leq n}$ of canonical models and $\mathfrak{D}_{\mathcal{K}} := (\mathcal{D}_{\mathcal{K}_i})_{0\leq i\leq n}$ of finite interpretations. Due to our assumption that each $\mathcal{I}_{\mathcal{K}_i}$ is countably infinite, and Condition (ii) of Definition 2.5, we can without loss of generality assume that these canonical models have the same domain. Similarly, the finite interpretations $\mathcal{D}_{\mathcal{K}_i}$ have the common domain $\Delta_{\mathcal{T}}$. Thus, they are valid sequences of interpretations according to our semantics (see Definition 3.1).

Finally, the temporal $\mathcal{Q}_2$-query $\phi^{\mathcal{T}}$ is obtained by replacing every $\mathcal{Q}_1$-query ψ occurring in ϕ by the $\mathcal{Q}_2$-query $\psi^{\mathcal{T}}$. We now obtain the following rewritability result, the proof of which can be found in Appendix A.

Theorem 4.1. *Let $\mathcal{Q}_1$, $\mathcal{Q}_2$ be query languages and $\mathcal{L}$ be a logic that has the canonical model property w.r.t. $\mathcal{Q}_1$ such that $\mathcal{Q}_1$-queries are $\mathcal{Q}_2$-rewritable w.r.t. $\mathcal{L}$. Then, for every consistent TKB $\mathcal{K} = \langle(\mathcal{A}_i)_{0\leq i\leq n}, \mathcal{T}\rangle$, every temporal $\mathcal{Q}_1$-query ϕ, and every i, $0 \leq i \leq n$, we have*

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

Our approach to answer temporal queries over data gathered while monitoring a system can thus be summarized as follows. Assume that we have an infinite TKB $\mathcal{K} = \langle(\mathcal{A}_i)_{i\geq 0}, \mathcal{T}\rangle$ that represents the sensor data coming from our system. At each time point $n \geq 0$, we only see the finite prefix $\mathcal{K}^{(n)} = \langle(\mathcal{A}_i)_{0\leq i\leq n}, \mathcal{T}\rangle$ of $\mathcal{K}$ of length $n+1$. In every step, we gain access to a new fact base $\mathcal{A}_{n+1}$ representing the sensor data of the current time point. Recall that $\mathcal{T}$ formalizes the fixed domain knowledge that holds at every time point. We now want to answer a fixed query ϕ, formulated in a query language $\mathcal{Q}_1$, at each time point.

Following the approach detailed above, we rewrite ϕ into a $\mathcal{Q}_2$-query $\phi^{\mathcal{T}}$. This can be done *offline*, i.e., before the system is started, since it does not depend on any sensor data. However, in each step, we have to construct the finite interpretation $\mathcal{D}_{\mathcal{K}_{n+1}}$ from $\mathcal{A}_{n+1}$ and $\mathcal{T}$ in order to extend the sequence $\mathfrak{D}_{\mathcal{K}^{(n)}}$. It now remains to show how to compute $\mathsf{Ans}(\phi^{\mathcal{T}}, \mathfrak{D}_{\mathcal{K}^{(n)}})$ in each step.

Since from now on we only need to consider the single query language $\mathcal{Q}_2$ and it does not matter how we obtained the query and the sequence of finite interpretations, we restate the problem in terms of a generic $\mathcal{Q}$-query and arbitrary finite interpretations.

Definition 4.2. Let $\mathfrak{I} = (\mathcal{I}_i)_{i\geq 0}$ an infinite sequence of interpretations over the finite domain Δ and ϕ be a temporal $\mathcal{Q}$-query. For every $n \geq 0$, we denote by $\mathfrak{I}^{(n)} = (\mathcal{I}_i)_{0\leq i\leq n}$ the finite prefix of $\mathfrak{I}$ of length $n+1$. The *temporal database monitoring problem* is the problem of computing the sequence $(\mathsf{Ans}(\phi, \mathfrak{I}^{(n)}))_{n\geq 0}$.

For simplicity, we assume that $\mathsf{N_C} = \Delta$ and $c^{\mathcal{I}_i} = c$ for all $c \in \mathsf{N_C}$, which can always be accomplished by introducing additional constants. This does not affect the semantics of the queries due to Conditions (i) and (ii) of Definition 2.5. Thus, in the following we regard answers to queries ϕ as mappings from $\mathsf{FVar}(\phi)$ to Δ. This is closer to the reading of the interpretations $\mathcal{I}_i$ as databases as, in this setting, one usually queries over all objects present in the database.

5. Solving the Temporal Database Monitoring Problem

We now illustrate two approaches to solving the temporal database monitoring problem on the small instance

$$\phi_{\mathsf{ex}} := \psi_a \wedge \psi_b \wedge \big(\psi_c\, \mathsf{S}(\bigcirc(\psi_b \wedge \bigcirc\Diamond\psi_b))\big)$$

of the introductory example, using the atemporal queries

$$\psi_a := \mathsf{TCS}(x) \wedge \mathsf{Server}(y) \wedge \mathsf{executes}(y, x);$$
$$\psi_b := \mathsf{Active}(x) \wedge \mathsf{Overloaded}(y);$$
$$\psi_c := \mathsf{NLB}(y).$$

Furthermore, we consider the subqueries $\phi_1 := \psi_c\,\mathsf{S}\,\phi_2$, $\phi_2 := \bigcirc(\psi_b \wedge \phi_3)$, and $\phi_3 := \bigcirc\Diamond\psi_b$. Since we have dispensed with knowledge bases in the previous section, we view ϕ_{ex} as a temporal query whose atoms are simple instance queries over database relations.

In the following examples, we consider the first five time points of a sequence $\mathfrak{I} = (\mathcal{I}_i)_{i\geq 0}$ of interpretations over the common domain $\Delta := \{s, p_1, p_2, p_3\}$. We define $\mathsf{TCS}^{\mathcal{I}_i} := \{p_1, p_2, p_3\}$, $\mathsf{Server}^{\mathcal{I}_i} := \{s\}$, $\mathsf{NLB}^{\mathcal{I}_i} := \{s\}$, and $\mathsf{executes}^{\mathcal{I}_i} := \{(s, p_1), (s, p_2), (s, p_3)\}$ for all time points i, and thus the sets of answers to ψ_c and ψ_a are always $\{s\}$ and $\{(p_1, s), (p_2, s), (p_3, s)\}$, respectively. We interpret the remaining predicates as in the following table, which results in the below listed answers to ψ_b:

i	$\mathsf{Active}^{\mathcal{I}_i}$	$\mathsf{Overloaded}^{\mathcal{I}_i}$	$\mathsf{Ans}(\psi_b, \mathcal{I}_i)$
0	$\{p_1, p_2\}$	$\emptyset$	$\emptyset$
1	$\{p_1, p_2, p_3\}$	$\{s\}$	$\{(p_1, s), (p_2, s), (p_3, s)\}$
2	$\{p_1, p_3\}$	$\emptyset$	$\emptyset$
3	$\{p_2, p_3\}$	$\{s\}$	$\{(p_2, s), (p_3, s)\}$
4	$\{p_3\}$	$\{s\}$	$\{(p_3, s)\}$

5.1. Temporal Database Query Languages

A first possibility to solve the temporal database monitoring problem is to cast $\mathfrak{I}$ as a temporal relational database and rewrite ϕ into a temporal database query language, in case this is possible. This works, for example, whenever $\mathcal{Q}$ contains only first-order queries, which can be expressed as SQL queries [1]. We illustrate this approach on the recursive translation from temporal logic to ATSQL described in [15]. For details on the syntax of ATSQL and the formal translation, see [15, 51].

ATSQL was developed for data annotated with time periods [51], and the approach from [15] works on *valid-time* periods that are required to always be *coalesced*, which means that they represent maximal, non-overlapping periods of time in which the data is valid. For example, the relation Active from our example would be represented in such a database by the tuples $(p_1, [0, 2])$, $(p_2, [0, 1])$, $(p_2, [3, 3])$, and $(p_3, [1, 4])$ consisting of transcoding services and the periods of time in which they are active.

In the following, we denote by $\mathsf{Q}(\phi)$ the ATSQL translation of a TQ ϕ. The atemporal queries are translated into standard SQL queries, for which the valid-time periods are automatically aggregated from the individual database tables by the database system. Likewise, $\mathsf{Q}(\phi_{\mathsf{ex}})$ can be computed as a simple join of $\mathsf{Q}(\psi_a)$, $\mathsf{Q}(\psi_b)$, and $\mathsf{Q}(\phi_1)$, and similarly for $\mathsf{Q}(\psi_b \wedge \phi_3)$. We now present the translation of the temporal formulae, which differs slightly from that in [15] because we use a different temporal semantics.

The ATSQL query $\mathsf{Q}(\phi_3)$ is quite simple:

```
NSEQ VT
SET VT PERIOD(0, END(VTIME(b))-1)
SELECT x, y FROM Q(ψb)(VT) as b
WHERE END(VTIME(b)) >= 1
```

The keyword `NSEQ VT` (for *non-sequential valid-time*) indicates that we want to modify the valid-time periods of the tuples in $\mathsf{Q}(\psi_b)$ (via `SET VT`), in contrast to `SEQ VT` (*sequential valid-time*), which tries to compute them automatically from the input tables. Consider now any answer tuple (x, y) of $\mathsf{Q}(\psi_b)$. The associated valid-time period $[i, j]$ can be accessed in an ATSQL query via the operator `VTIME`. The valid-time period of (x, y) in $\mathsf{Q}(\phi_3)$ is then computed as $[0, j-1]$ since $\phi_3 = \bigcirc\Diamond\psi$ is true iff there is a point in the future (different from the current time point) where ψ is true. In contrast to [15], where the temporal dimension starts with $-\infty$, for us the first time point is 0. The keyword `(VT)` in the `FROM` clause enforces the coalescing of the tuples from $\mathsf{Q}(\psi_b)$. By likewise coalescing the result of $\mathsf{Q}(\phi_3)$, we obtain three answer tuples:

x	y	$[i,j]$	$[0,j-1]$	coalesced
p_1	s	$[1,1]$	$[0,0]$	$[0,0]$
p_2	s	$[1,1]$	$[0,0]$	$[0,2]$
p_2	s	$[3,3]$	$[0,2]$	
p_3	s	$[1,1]$	$[0,0]$	$[0,3]$
p_3	s	$[3,4]$	$[0,3]$	

The ATSQL translation of ϕ_2 is

```
NSEQ VT
SET VT PERIOD(LAST(0,BEGIN(VTIME(b))-1),
              END(VTIME(b))-1)
SELECT x, y FROM Q(ψ_b ∧ φ_3)(VT) as b
WHERE END(VTIME(b)) >= 1
```

This query shifts the answers to $\mathsf{Q}(\psi_b \wedge \phi_3)$ by one time step, except when this would result in negative time points. We obtain the tuples $(p_2, s, [0,0])$, $(p_3, s, [0,0])$, and $(p_3, s, [2,2])$.

We next compute the auxiliary query $\mathsf{Q}_{\mathsf{aux}}$, which is a join of $\mathsf{Q}(\psi_c)$ and $\mathsf{Q}(\phi_2)$ that explicitly retains the valid-time periods of the two subqueries:

```
NSEQ VT
SELECT b.x, b.y, VTIME(c) as p1,
       VTIME(b) as p2
FROM Q(ψ_c)(VT) as c, Q(φ_2)(VT) as b
WHERE c.y = b.y
```

The result of this query is now used in $\mathsf{Q}(\phi_1)$ as follows:

```
(SET VT PERIOD(END(p2)+1, END(p1))
 SELECT x, y FROM Q_aux as aux
 WHERE END(p2)+1 >= BEGIN(p1)
 AND END(p1) >= END(p2)+1)
UNION
(SET VT p2
 SELECT x, y FROM Q_aux as aux)
```

Intuitively, the query ϕ_1 collects, for each combination of the variables x and y, all periods from $\mathsf{Q}(\phi_2)$ (since there the S-formula is immediately satisfied), together with the last part of those periods from $\mathsf{Q}(\psi_c)$ that meet or overlap the end of a matching period from $\mathsf{Q}(\phi_2)$. By *matching* we mean that the values of the shared variable coincide (`c.y = b.y`). After coalescing, the resulting tuples are $(p_2, s, [0,4])$ and $(p_3, s, [0,4])$. Intersecting these with the answers for $\mathsf{Q}(\psi_a \wedge \psi_b)$, we obtain $(p_2, s, [1,1])$, $(p_2, s, [3,3])$, $(p_3, s, [1,1])$, and $(p_3, s, [3,4])$.

Since we are only interested in the answers for the last time point 4 (until new data arrives), this results in a warning that p_3 is currently active while s is overloaded, and this situation has happened at least once before since the last load balancing operation. At the previous time point 3, a warning was issued for both p_2 and p_3. In contrast, at time point 1 only the data from $\mathcal{I}_0$ and $\mathcal{I}_1$ was available, and thus no warning was issued.

This translation illustrates the advantage of using valid-time *periods* instead of individual time points, as we only have to simply manipulate the endpoints of the periods. However, since our goal is to monitor systems that produce new data in very short time intervals, storing all past data, even compressed into periods, is not feasible.

5.2. Bounded History Encodings

In the remainder of this paper, we describe two different approaches that reduce the amount of space necessary to compute $\mathsf{Ans}(\phi, \mathfrak{I}^{(n)})$. 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 TQ ϕ. This is formalized by the notion of a *bounded history encoding* in [16, 18].

Definition 5.1 (history encoding). Given a TQ ϕ, a *history encoding* for ϕ is a tuple $(\Delta^{\mathcal{E}}, I^{\mathcal{E}}, \delta^{\mathcal{E}}, \phi^{\mathcal{E}})$, where $\Delta^{\mathcal{E}}$ is the set of *encodings*, $I^{\mathcal{E}} \in \Delta^{\mathcal{E}}$ is the *initial encoding*, $\delta^{\mathcal{E}} \colon \Delta^{\mathcal{E}} \times \mathfrak{F} \to \Delta^{\mathcal{E}}$ is the *transition function* (where $\mathfrak{F}$ denotes the set of all finite interpretations), and $\phi^{\mathcal{E}} \colon \Delta^{\mathcal{E}} \to 2^{\Delta^{\mathsf{FVar}(\phi)}}$ is the *evaluation function.* This tuple defines an operator $\mathcal{E}$ mapping finite sequences $\mathfrak{I}^{(n)} = (\mathcal{I}_i)_{0 \le i \le n}$ of finite interpretations over the same domain to encodings in $\Delta^{\mathcal{E}}$ as follows: $\mathcal{E}(()) := I^{\mathcal{E}}$, and $\mathcal{E}(\mathfrak{I}^{(n)}) := \delta^{\mathcal{E}}(\mathcal{E}(\mathfrak{I}^{(n-1)}), \mathcal{I}_n)$ for all $n \ge 0$. It is *correct* if we have $\mathsf{Ans}(\phi, \mathfrak{I}^{(n)}) = \phi^{\mathcal{E}}(\mathcal{E}(\mathfrak{I}^{(n)}))$ for all $\mathfrak{I}^{(n)}$, $n \ge 0$. It is *bounded* if the size of $\mathcal{E}(\mathfrak{I}^{(n)})$ does not depend on the length n of the history.

Note that history encodings are called *expiration operators* in [18]. Whenever new data arrives in the form of a finite interpretation $\mathcal{I}_n$, the previously computed encoding $\mathcal{E}(\mathfrak{I}^{(n-1)})$ is updated via the function $\delta^{\mathcal{E}}$. Correctness is an obvious requirement for any encoding since we still want to be able to answer the original TQ after encoding the data. The boundedness condition ensures that the space required to answer the query does not depend on the number n of previous time points; only the relevant data from the past is retained (in aggregated form).

Note that the approach of Section 5.1 constitutes a history encoding: the encoding of a sequence of interpretations is the corresponding temporal database with valid-time periods, and the evaluation function is given by the translation into ATSQL sketched above. This history encoding is correct, but obviously not bounded.

In the following, we describe two possible methods to achieve a *bounded* history encoding. In the first approach (Section 5.3), we rewrite ϕ into a TQ ϕ' without future operators by employing a result from [52]. We then compute $\mathsf{Ans}(\phi', \mathfrak{I}^{(n)})$ via a bounded history encoding described in [16, 18]. In Section 6, we generalize the algorithm from [16, 18] to directly deal with future operators. The main difference is that we do not consider negation or arbitrary first-order temporal queries. This allows us to circumvent the non-elementary blowup of the formula resulting from the reduction in [52], while retaining boundedness.

5.3. Eliminating Future Operators

In this section, we show that we can rewrite every temporal query ϕ into an equivalent TQ ϕ' that does not contain future operators but may contain negation as in [16]. We then apply the algorithm described in [16] to iteratively compute the sets $\mathsf{Ans}(\phi', \mathfrak{I}^{(n)})$.

The reduction proceeds in the following steps. First, we transform ϕ into a (temporally) equivalent propositional LTL-formula in order to then apply the separation theorem from [52]. This produces a propositional LTL-formula in which no future operators occur in the scope of past operators and vice versa. Since we evaluate the query at the current (last) time point, this allows us to simply remove the future operators. Finally, the resulting formula is translated back into a TQ extended with negation.

For the first translation, note that our temporal semantics differs from that in [52], which considers strict versions of U and S as the only temporal operators. But it is well-known that these operators can simulate $\bigcirc$ and $\bigcirc^-$. Moreover, the semantics is defined w.r.t. bounded past and unbounded future.

Definition 5.2 (Propositional LTL). 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 $\mathfrak{J} = (w_i)_{i\geq 0}$ of *worlds* $w_i \subseteq P$, $i \geq 0$, and it *satisfies* an LTL-formula ϕ at $i \geq 0$ if $\mathfrak{J}, i \models \phi$ holds, which is defined inductively:

ϕ	$\mathfrak{J}, i \models \phi$ iff
$p \in P$	$p \in w_i$
$\phi_1 \wedge \phi_2$	$\mathfrak{J}, i \models \phi_1$ and $\mathfrak{J}, i \models \phi_2$
$\phi_1 \vee \phi_2$	$\mathfrak{J}, i \models \phi_1$ or $\mathfrak{J}, i \models \phi_2$
$\neg\phi_1$	not $\mathfrak{J}, i \models \phi_1$
$\phi_1 \,\mathsf{U}^<\, \phi_2$	there is some $k > i$ with $\mathfrak{J}, k \models \phi_2$ and $\mathfrak{J}, j \models \phi_1$ for all j, $i < j < k$
$\phi_1 \,\mathsf{S}^<\, \phi_2$	there is some k, $0 \leq k < i$, with $\mathfrak{J}, k \models \phi_2$ and $\mathfrak{J}, j \models \phi_1$ for all j, $k < j < i$

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 $\mathfrak{J}, i \models \mathsf{first}$ iff $i = 0$, i.e., this formula is satisfied exactly at the first time point.

Let from now on ϕ be an arbitrary but fixed TQ containing only the $\mathcal{Q}$-queries $\psi_1, \ldots, \psi_m$. 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 and a variable assignment $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \mathsf{N_C}$, the *propositional abstraction* is the LTL-structure $\mathfrak{J}_\mathfrak{a} := (w_i)_{i\geq 0}$, where

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

There, the propositional variables p_i, $1 \leq i \leq m$, capture whether $\mathfrak{a}$ is an answer to ψ_i. The additional variable p is used to distinguish the first n time points. This is necessary since the semantics of TQs considers only the first n time points whereas in LTL all time points matter.

The first step of the translation yields an LTL-formula f_ϕ that behaves similarly to ϕ w.r.t. the propositional abstractions of sequences of interpretations $\mathfrak{I}$ and variable assignments $\mathfrak{a}$. The formal construction is shown in Appendix B; we only illustrate it here on the example of ϕ_{ex}. Assume that the propositional variables p_a, p_b, p_c are used for ψ_a, ψ_b, ψ_c, respectively. Then, the corresponding formula $f_{\phi_{\mathsf{ex}}}$ looks as follows:

$$f_{\phi_{\mathsf{ex}}} := p_a \wedge p_b \wedge \big(f_{\phi_2} \vee (p_c \wedge p_c \,\mathsf{S}^<\, f_{\phi_2})\big)$$

where $f_{\phi_2} := \mathsf{false}\,\mathsf{U}^<(p_b \wedge f_{\phi_3} \wedge p)$ and $f_{\phi_3} := \mathsf{true}\,\mathsf{U}^<(p_b \wedge p)$. The main differences to the temporal structure of ϕ_{ex} are that the non-strict S is simulated using the strict version and the future operators are simulated via $\mathsf{U}^<$.

We now use the separation theorem from [52] to transform f_ϕ into an equivalent LTL-formula f'_ϕ that is a Boolean combination of temporal subformulae containing 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.

In our example, only the subformula $p_c \,\mathsf{S}^<\, f_{\phi_2}$ is not yet separated. Its separation according to the transformation in [52] is the disjunction of the following formulae:

- $p_b \wedge f_{\phi_3} \wedge p \wedge \mathsf{false}\,\mathsf{S}^<\,\mathsf{true} \wedge p_c \,\mathsf{S}^<\,\mathsf{true}$
- $p_c \,\mathsf{S}^<\, \chi_1 \wedge \mathsf{true}\,\mathsf{S}^<\, \chi_1 \wedge \mathsf{true}\,\mathsf{U}^<(p_b \wedge p)$
- $p_b \wedge p \wedge \mathsf{true}\,\mathsf{S}^<\, \chi_1 \wedge p_c \,\mathsf{S}^<\, \chi_1$
- $p_c \,\mathsf{S}^<(p_b \wedge p \wedge p_c \wedge \mathsf{true}\,\mathsf{S}^<\, \chi_1 \wedge p_c \,\mathsf{S}^<\, \chi_1)$

where

$$\chi_1 := p_b \wedge p \wedge p_c \wedge \mathsf{false}\,\mathsf{S}^<\,\mathsf{true} \wedge p_c \,\mathsf{S}^<\,\mathsf{true}.$$

This is obtained by a case analysis of the possible relations between the time intervals covered by the $\mathsf{S}^<$ operator and the two $\mathsf{U}^<$ operators in f_{ϕ_2}.

We simplify this formula for the subsequent constructions. Note that χ_1 is equivalent to

$$\chi_2 := p_b \wedge p \wedge p_c \wedge \neg\mathsf{first}$$

and the first disjunct is equivalent to $f_{\phi_3} \wedge \chi_2$. Since $p_c \,\mathsf{S}^<\, \chi_1$ implies $\mathsf{true}\,\mathsf{S}^<\, \chi_1$, we have

$$f'_{\phi_{\mathsf{ex}}} := p_a \wedge p_b \wedge \big(f_{\phi_2} \vee (p_c \wedge \zeta_1)\big)$$

where ζ_1 is the disjunction of the following formulae:

- $f_{\phi_3} \wedge \chi_2$
- $p_c \,\mathsf{S}^<\, \chi_2 \wedge \mathsf{true}\,\mathsf{U}^<(p_b \wedge p)$
- $p_b \wedge p \wedge p_c \,\mathsf{S}^<\, \chi_2$

- $p_c \, \mathsf{S}^< (p_b \wedge p \wedge p_c \wedge p_c \, \mathsf{S}^< \chi_2)$.

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 an $\mathsf{U}^<$ 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 standard equivalences as shown in Appendix B. This yields a formula f''_ϕ that does not contain any $\mathsf{U}^<$ operators and is equivalent to f'_ϕ at time point n in every LTL-structure of the form $\mathfrak{I}_\mathfrak{a}$.

In our example, we obtain $f''_{\phi_{\mathsf{ex}}} := p_a \wedge p_b \wedge p_c \wedge \zeta_2$ with

$$\zeta_2 := (p_b \wedge p \wedge p_c \, \mathsf{S}^< \chi_2) \vee (p_c \, \mathsf{S}^< (p_b \wedge p \wedge p_c \wedge p_c \, \mathsf{S}^< \chi_2)).$$

We now translate the LTL-formula f''_ϕ without $\mathsf{U}^<$ back into a TQ $\phi_{f''_\phi}$. Recall that the goal is to use the algorithm presented in [16], 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 [16], is as follows:

ϕ	$\mathfrak{I}, i \models \mathfrak{a}(\phi)$ iff
$\neg\phi_1$	not $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$
$\phi_1 \, \mathsf{S}^* \, \phi_2$	there is a k, $0 \leq k < i$, with $\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2)$ and $\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1)$ for all j, $k < j \leq i$

In the following, we call any TQ built using the operators $\wedge$, $\vee$, $\neg$, $\ocircle^-$, and S^* a *Past-TQ*, which is in particular a temporal query in the sense of [16]. The formal definition of this final translation to the Past-TQ $\phi_{f''_\phi}$ is given in Appendix B.

In our example, we obtain $\phi_{f''_{\phi_{\mathsf{ex}}}} := \psi_a \wedge \psi_b \wedge \psi_c \wedge \zeta_3$, where

$$\zeta_3 := (\psi_b \wedge \zeta_4) \vee \ocircle^-((\psi_b \wedge \psi_c \wedge \zeta_4) \vee \psi_c \, \mathsf{S}^*(\psi_b \wedge \psi_c \wedge \zeta_4))$$

and

$$\zeta_4 := \ocircle^-((\psi_b \wedge \psi_c \wedge \ocircle^-\mathsf{true}) \vee \psi_c \, \mathsf{S}^*(\psi_b \wedge \psi_c \wedge \ocircle^-\mathsf{true})).$$

Note that ψ_c occurs 13 times in $\phi_{f''_{\phi_{\mathsf{ex}}}}$, but only once in the original query ϕ_{ex}. While some copies where introduced because of the different semantics of S, $\mathsf{S}^<$, and S^*, the main problem in this translation is the separation theorem [52]. In general, the size of the separated formula may be non-elementary in the size of the original formula; the number of stacked exponents is determined by the number of alternations between nested $\mathsf{S}^<$ and $\mathsf{U}^<$ operators.

Taken together, the illustrated translations yield the following result, which is proven in Appendix B.

Theorem 5.3. *For every TQ ϕ, there is a Past-TQ ψ with $\mathsf{FVar}(\phi) = \mathsf{FVar}(\psi)$ such that 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 solve the temporal database monitoring problem using the bounded history encoding from [16], which works as follows on the TQ ψ constructed in Theorem 5.3.

The encodings consist of a finite interpretation $\mathcal{I}'_i$ of several auxiliary predicates. Intuitively, for each subformula ψ' of ψ starting with a past operator, it stores the answers $\mathsf{Ans}(\psi', \mathfrak{I}^{(i)}) \subseteq \Delta^{\mathsf{FVar}(\psi')}$ for ψ' at the current time point i. The set $\mathsf{Ans}(\psi, \mathfrak{I}^{(i)})$ can then easily be computed from the current interpretation $\mathcal{I}_i$ and $\mathcal{I}'_i$, i.e., the construction yields a correct history encoding. Afterwards, $\mathcal{I}_i$ is disregarded and the information computed in $\mathcal{I}'_i$ is the only one kept. On input $\mathcal{I}_{i+1}$, the previous encoding $\mathcal{I}'_i$ is updated to a new interpretation $\mathcal{I}'_{i+1}$, which allows us to compute $\mathsf{Ans}(\psi, \mathfrak{I}^{(i+1)})$, and so on.

The size of $\mathcal{I}'_i$ is bounded polynomially in the size of Δ and in the number of past operators occurring in ψ, and exponentially in the number of free variables occurring below past operators. However, the memory requirements of this history encoding do not depend on n, and thus it is bounded.

Note that a formal requirement for the correctness of the algorithm in [16] is that ψ is *domain-independent*, which means that the answers to ψ at previous time points do not change if the domain is changed from the current time point to the next (e.g., by introducing new constants). Otherwise, the answers to the past formulae at the current time point could not be compiled into a single interpretation $\mathcal{I}'_i$ so easily, 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 only dealing with the constant, finite domain $\Delta_\mathcal{T} = \mathsf{N_C}$ (see Section 4), we do not need to assume domain-independence of ψ.

The approach presented in this section has the obvious drawback that the reduction in [52] is non-elementary in the size of the formula. As mentioned before, for propositional LTL, eliminating the past operators from a formula incurs at least an exponential blowup; the best known construction works via translation through several logics and automata models, and is therefore also hardly practical [5]. The main advantage arises from the fact that the approach described in [16] can easily be implemented in a standard database system. No temporal information needs to be stored and only several auxiliary tables have to be updated after new sensor information becomes available.

6. Bounded History Encodings for Future Operators

In this section, we present an algorithm that solves the temporal database monitoring problem without the need to eliminate the future operators from the query, thereby avoiding the non-elementary blowup of the construction described in the previous section. We further show that this approach also constitutes a bounded history encoding.

As before, let ϕ be a fixed temporal $\mathcal{Q}$-query over some query language $\mathcal{Q}$ for which answers w.r.t. finite interpretations are computable, and let $\mathfrak{I} = (\mathcal{I}_i)_{i\geq 0}$ be a fixed infinite sequence of interpretations over the same finite domain Δ. For ease of presentation, we do not consider the temporal operators $\Box$, $\Diamond$, $\Box^-$, and $\Diamond^-$ in this section. The constructions and arguments for these operators are similar to those for U and S.

The algorithm uses as data structure so-called *answer terms*, which represent TQs in which some parts have already been evaluated. In particular, they do not contain atemporal queries anymore, 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 simplicity, 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 $\mathsf{Ans}(\phi, \mathfrak{I}^{(n)})$ refers to a set of such mappings, i.e., a subset of $\Delta^{\mathsf{N_V}}$.

The domain of our bounded history encoding essentially consists of (families of) answer terms, as defined next.

Definition 6.1 (answer term). Let $\mathsf{FSub}(\phi)$ denote the set of all subqueries of ϕ of the form $\bigcirc\psi_1$, $\bullet\psi_1$, or $\psi_1 \mathbin{\mathsf{U}} \psi_2$. 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 AT_ϕ^i of all *answer terms 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 term for ϕ at i.
- Every variable $x_j^\psi \in \mathsf{Var}_j^\phi$ with $j \leq i$ is an answer term for ϕ at i.
- If α_1 and α_2 are answer terms for ϕ at i, then so are $\alpha_1 \cap \alpha_2$ and $\alpha_1 \cup \alpha_2$.

Note that every answer term at i is also an answer term at $i+1$, i.e., we have $\mathsf{AT}_\phi^i \subseteq \mathsf{AT}_\phi^{i+1}$.

We now define an evaluation function, mapping answer terms to sets of answers. Intuitively, it replaces the variables x_j^ψ in α by appropriate sets of answers and evaluates $\cap$ and $\cup$ as set intersection and union, respectively. Formally, the functions $\mathsf{eval}^n\colon \mathsf{AT}_\phi^n \to 2^{\Delta^{\mathsf{N_V}}}$, $n \geq 0$, are defined recursively as follows:

α	$\mathsf{eval}^n(\alpha)$
$A \subseteq \Delta^{\mathsf{N_V}}$	A
$x_j^{\bigcirc\psi_1}$ with $j < n$	$\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, j+1)$
$x_j^{\bullet\psi_1}$ with $j < n$	$\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, j+1)$
$x_j^{\psi_1 \mathsf{U} \psi_2}$ with $j < n$	$\mathsf{Ans}(\psi_1 \mathbin{\mathsf{U}} \psi_2, \mathfrak{I}^{(n)}, j+1)$
$x_n^{\bigcirc\psi_1}$	$\emptyset$
$x_n^{\bullet\psi_1}$	$\Delta^{\mathsf{N_V}}$
$x_n^{\psi_1 \mathsf{U} \psi_2}$	$\emptyset$
$\alpha_1 \cap \alpha_2$	$\mathsf{eval}^n(\alpha_1) \cap \mathsf{eval}^n(\alpha_2)$
$\alpha_1 \cup \alpha_2$	$\mathsf{eval}^n(\alpha_1) \cup \mathsf{eval}^n(\alpha_2)$

As mentioned before, our data structure consists of *families* $\Phi\colon \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i$ of answer terms, one for each subquery of ϕ. This additional information is necessary for the transition function of our encoding. We would actually only need to store the answer terms for subqueries that are referred to by past operators, but for simplicity, we compute them here for all subqueries.

For our proofs, we also need more fine-grained notions of correctness and boundedness.

Definition 6.2 (correct, bounded). Consider a function $\Phi\colon \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i$. This function is *correct for $i \geq 0$* (w.r.t. $\mathfrak{I}$) if for all $n \geq i$ and for all $\psi \in \mathsf{Sub}(\phi)$, we have $\mathsf{eval}^n(\Phi(\psi)) = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)$. It is *$i$-bounded* if for all $\psi \in \mathsf{Sub}(\phi)$, the answer term $\Phi(\psi)$ contains only variables from Var_i^ψ.

In particular, if $\Phi\colon \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i$ is correct for i, then we can compute the desired set $\mathsf{Ans}(\phi, \mathfrak{I}^{(i)})$ by evaluating $\mathsf{eval}^i(\Phi(\phi))$, and thus our encoding is valid in the sense of Definition 5.1. Moreover, if Φ is i-bounded, then $\mathsf{eval}^i(\Phi(\phi))$ can easily be computed since all variables simply have to be replaced with $\emptyset$ or $\Delta^{\mathsf{N_V}}$.

We now define the initial encoding Φ_0 and prove that it is correct for 0. Afterwards, we define the transition function that computes a new encoding Φ_i from the previous encoding Φ_{i-1} and the next interpretation $\mathcal{I}_i$, and then prove that it preserves correctness.

6.1. Computing the Answer Terms

The answer terms $\Phi_0(\psi)$, $\psi \in \mathsf{Sub}(\phi)$, are defined by induction on the structure of TQs as follows:

ψ	$\Phi_0(\psi)$
atemporal query ψ_1	$\mathsf{Ans}(\psi_1, \mathcal{I}_0)$
$\psi_1 \wedge \psi_2$	$\Phi_0(\psi_1) \cap \Phi_0(\psi_2)$
$\psi_1 \vee \psi_2$	$\Phi_0(\psi_1) \cup \Phi_0(\psi_2)$
$\bigcirc\psi_1$	$x_0^{\bigcirc\psi_1}$
$\bigcirc^-\psi_1$	$\emptyset$
$\bullet\psi_1$	$x_0^{\bullet\psi_1}$
$\bullet^-\psi_1$	$\Delta^{\mathsf{N_V}}$
$\psi_1 \mathbin{\mathsf{U}} \psi_2$	$\Phi_0(\psi_2) \cup (\Phi_0(\psi_1) \cap x_0^{\psi_1 \mathsf{U} \psi_2})$
$\psi_1 \mathbin{\mathsf{S}} \psi_2$	$\Phi_0(\psi_2)$

To answer atemporal queries ψ_1, we employ our assumption that the set $\mathsf{Ans}(\psi_1, \mathcal{I}_0)$ is computable since $\mathcal{I}_0$ is a finite interpretation. We can thus use any known algorithm to compute this set, which is then considered as an atom of the answer term $\Phi_0(\phi)$ for ϕ.

To understand the idea behind the variables x_i^{ψ}, consider for example $x_0^{\bigcirc\psi_1}$, which is the answer term for $\bigcirc\psi_1$ at time point 0. This variable serves as a place-holder for the answers to ψ_1 at the next time point 1, which are not yet known. Thus, according to the semantics, the set of answers to $\bigcirc\psi_1$ at time point 0 is $\emptyset$, which is equal to $\mathsf{eval}^0(x_0^{\bigcirc\psi_1})$. However, once data about time point 1 becomes known, the variable $x_0^{\bigcirc\psi_1}$ can be substituted by the actual answers to ψ_1 at time point 1. For more details on this substitution, see the proof of Lemma 6.5 in Appendix C.

Similarly, $x_0^{\psi_1 \mathsf{U} \psi_2}$ is a place-holder for the answers to $\psi_1 \mathsf{U} \psi_1$ at time point 1. This means that we simply evaluate the U operator according to the equivalence of $\psi_1 \mathsf{U} \psi_2$ and $\psi_2 \vee (\psi_1 \wedge \bigcirc(\psi_1 \mathsf{U} \psi_2))$ (cf. Proposition 3.4).

The following result is proven in Appendix C.

Lemma 6.3. *The function Φ_0 is correct for* 0.

Furthermore, Φ_0 is obviously 0-bounded. Assume now that $i > 0$ and we are given a function $\Phi_{i-1} \colon \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^{i-1}$ that is correct for $i-1$ and $(i-1)$-bounded. We now describe the transition function that computes a new function Φ_i that is correct for i and i-bounded, using the data from the next interpretation $\mathcal{I}_i$.

As a first step, we define a function $\Phi_i^0 \colon \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i$ (similar to Φ_0) that is correct for i, but may still contain variables with index $i-1$. Afterwards, we appropriately replace these variables while ensuring that correctness for i is preserved.

For $i > 0$ and given $\Phi_{i-1} \colon \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^{i-1}$, the mapping $\Phi_i^0 \colon \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i$ is defined recursively as follows:

ψ	$\Phi_i^0(\psi)$
atemporal query ψ_1	$\mathsf{Ans}(\psi_1, \mathcal{I}_i)$
$\psi_1 \wedge \psi_2$	$\Phi_i^0(\psi_1) \cap \Phi_i^0(\psi_2)$
$\psi_1 \vee \psi_2$	$\Phi_i^0(\psi_1) \cup \Phi_i^0(\psi_2)$
$\bigcirc\psi_1$	$x_i^{\bigcirc\psi_1}$
$\bigcirc^-\psi_1$	$\Phi_{i-1}(\psi_1)$
$\bullet\psi_1$	$x_i^{\bullet\psi_1}$
$\bullet^-\psi_1$	$\Phi_{i-1}(\psi_1)$
$\psi_1 \mathsf{U} \psi_2$	$\Phi_0(\psi_2) \cup (\Phi_i^0(\psi_1) \cap x_i^{\psi_1 \mathsf{U} \psi_2})$
$\psi_1 \mathsf{S} \psi_2$	$\Phi_i^0(\psi_2) \cup (\Phi_i^0(\psi_1) \cap \Phi_{i-1}(\psi_1 \mathsf{S} \psi_2))$

The difference to the definition of Φ_0 is that the answer terms for past operators are computed using the answer terms for the previous time point. Correctness of this mapping is shown in Appendix C.

Lemma 6.4. *If Φ_{i-1} is correct for $i-1$, then Φ_i^0 is correct for i.*

In order to remove the 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 $\Phi_i^0(\psi)$. The details of this construction are described in Appendix C.

Lemma 6.5. *If Φ_{i-1} is correct for $i-1$ and $(i-1)$-bounded, then we can construct a function $\Phi_i \colon \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i$ that is correct for i and i-bounded.*

To solve the temporal database monitoring problem on input ϕ and $\mathfrak{I}$, we now iteratively compute the mappings Φ_i as described above, and output $\mathsf{eval}^i(\Phi_i(\phi))$ for each $i \geq 0$. By the previous lemmata, this is a correct history encoding.

If the query ϕ contains no future operators, then the answer terms 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 the size of $\Phi_i(\phi)$ is independent of i, and thus we have constructed a bounded history encoding, which can be seen as a variant of the one from [16, 18].

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. Unfortunately, the property of i-boundedness alone does not suffice since the variables from Var_i^ϕ may still occur several times in $\Phi_i(\phi)$.

Example 6.6. Consider again the query

$$\phi_{\mathsf{ex}} = \psi_a \wedge \psi_b \wedge \big(\mathsf{NLB}(y)\, \mathsf{S} (\bigcirc(\psi_b \wedge \bigcirc\Diamond\psi_b))\big)$$

from Section 5 and recall the abbreviations ϕ_1, ϕ_2, and ϕ_3 for the temporal subqueries of ϕ_{ex}. The answer terms for each time point i can be obtained as

$$\Phi_i(\phi_{\mathsf{ex}}) = \mathsf{Ans}(\psi_a, \mathcal{I}_i) \cap \mathsf{Ans}(\psi_b, \mathcal{I}_i) \cap \Phi_i(\phi_1).$$

To compute $\Phi_i(\phi_1)$, observe first that, for $i > 0$,

$$\Phi_i^0(\phi_1) = x_i^{\phi_2} \cup \big(\mathsf{Ans}(\mathsf{NLB}(y), \mathcal{I}_i) \cap \Phi_{i-1}(\phi_1)\big).$$

Since we consider all subqueries to have the same variables, $\mathsf{Ans}(\mathsf{NLB}(y), \mathcal{I}_i)$ evaluates to $\Delta \times \{s\}$ for all $i \in \{0, \dots, 4\}$. Hence, in our example this set does not affect the computations, and thus we will omit it and consider $\Phi_i^0(\phi_1)$ to be the union of $x_i^{\phi_2}$ and the answer term for ϕ_1 from the previous time point.

We now describe how the algorithm proceeds in more detail. A summary of the answer terms (equivalent to) $\Phi_i(\phi_1)$, $i \in \{0, \dots, 4\}$, can be found in the following table, where B_i abbreviates $\mathsf{Ans}(\psi_b, \mathcal{I}_i)$:

i	$\Phi_i(\phi_1)$	$\mathsf{eval}^i(\Phi_i(\phi_{\mathsf{ex}}))$
0	$x_0^{\phi_2}$	$\emptyset$
1	$x_1^{\phi_2} \cup (B_1 \cap x_1^{\phi_3})$	$\emptyset$
2	$x_2^{\phi_2} \cup (B_1 \cap x_2^{\Diamond\psi_b})$	$\emptyset$
3	$x_3^{\phi_2} \cup B_3 \cup (B_3 \cap x_3^{\phi_3}) \cup (B_1 \cap x_3^{\Diamond\psi_b})$	B_3
4	$x_4^{\phi_2} \cup B_3 \cup (B_4 \cap x_4^{\phi_3}) \cup (B_1 \cap x_4^{\Diamond\psi_b})$	B_4

To obtain the answer sets $\mathsf{eval}^i(\Phi_i(\phi_{\mathsf{ex}}))$, observe that by the definition of eval^i all variables are replaced by $\emptyset$ since $\Diamond\psi_b$ is equivalent to $\mathsf{true}\,\mathsf{U}\,\psi_b$. There are no answers to ϕ_{ex} at the first three time points since the combination of S with the two $\bigcirc$ operators requires at least three previous time points to exist. However, at time points 3 and 4, we obtain the sets $B_3 = \{(p_2, s), (p_3, s)\}$ and $B_4 = \{(p_3, s)\}$, respectively, as expected.

The computation for $i = 0$ is straightforward. For $i = 1$, we first compute

$$\Phi_1^0(\phi_1) = x_1^{\phi_2} \cup \Phi_0(\phi_1) = x_1^{\phi_2} \cup x_0^{\phi_2}.$$

Afterwards, we replace $x_0^{\phi_2}$ by $\Phi_1^0(\psi_b \wedge \phi_3) = B_1 \cap x_1^{\phi_3}$ since $\phi_2 = \bigcirc(\psi_b \wedge \phi_3)$, i.e., ϕ_2 at time point 0 refers to $\psi_b \wedge \phi_3$ at time point 1 (see the proof of Lemma 6.5 for details). We thus obtain the 1-bounded answer term $\Phi_1(\phi_1) = x_1^{\phi_2} \cup (B_1 \cap x_1^{\phi_3})$ listed above.

At $i = 2$, we get $\Phi_2^0(\phi_1) = x_2^{\phi_2} \cup x_1^{\phi_2} \cup (B_1 \cap x_1^{\phi_3})$. By replacing the variables with index 1, we compute

$$\Phi_2(\phi_1) = x_2^{\phi_2} \cup (B_2 \cap x_2^{\phi_3}) \cup \big(B_1 \cap (B_2 \cup x_2^{\Diamond\psi_b})\big).$$

Since $B_2 = \emptyset$, one can obviously simplify this term. For example, $\emptyset \cap x_2^{\phi_3}$ cannot evaluate to a non-empty answer set, and $\emptyset \cup x_2^{\Diamond\psi_b}$ yields the same results as $x_2^{\Diamond\psi_b}$ itself.

Without these simplifications, at $i = 3$, we would compute $\Phi_3(\phi_1)$ as

$$x_3^{\phi_2} \cup (B_3 \cap x_3^{\phi_3}) \cup \big(B_2 \cap (B_3 \cup x_3^{\Diamond\psi_b})\big) \cup \big(B_1 \cap (B_2 \cup B_3 \cup x_3^{\Diamond\psi_b})\big),$$

which contains $x_3^{\Diamond\psi_b}$ twice. In general, in each step we would add one copy of $x_i^{\Diamond\psi_b}$ to the answer term, which would result in a correct history encoding for ϕ that is, however, not bounded.

Fortunately, by simplifying all answer terms using the properties of $\cap$ and $\cup$ and the fact that $B_4 \subseteq B_3 \subseteq B_1$, we can compute the ($i$-bounded) answer terms $\Phi_i(\phi_1)$ given in the table above.

This demonstrates that it is important that the computed answer terms are simplified at each step, while preserving their behavior under eval^i.

6.2. Simplifying the Answer Terms

We show how to automatically simplify every answer term by rewriting it into a certain normal form. While variables from Var_i^ϕ may still occur several times in this normal form, the number of their occurrences does not depend on the number n of previous time points.

Definition 6.7. Two answer terms $\alpha_1, \alpha_2 \in \mathsf{AT}_\phi^i$ are *equivalent* (at i) if $\mathsf{eval}^n(\alpha_1) = \mathsf{eval}^n(\alpha_2)$ holds for all $n \geq i$. An answer term $\alpha \in \mathsf{AT}_\phi^i$ is in *normal form* (for i) if it is of the form

$$\bigcup_{X \subseteq \mathsf{Var}_i^\phi} \Big(A_X \cap \bigcap_{x \in X} x\Big),$$

where $A_X \subseteq \Delta^{\mathsf{N_V}}$ for each $X \subseteq \mathsf{Var}_i^\phi$.

Note that every i-bounded answer term α can be transformed into an equivalent one in normal form. To this end, we first transform it into disjunctive normal form, which may cause an exponential blowup. We then combine all conjunctions containing the same combination X of variables from Var_i^ϕ, and merge the already computed sets of answers into one set A_X. If one combination X occurs in no conjunction, we set $A_X := \emptyset$. If X has no associated sets of answers, we set $A_X := \Delta^{\mathsf{N_V}}$. It is easy to see that the resulting answer term is equivalent to the original one.

Proposition 6.8. *For every i-bounded answer term we can construct an equivalent answer term that is in normal form for i.*

Consider for example the (non-simplified) answer term $\Phi_3(\phi_1)$ from Example 6.6 above. An equivalent term in disjunctive normal form is

$$x_3^{\phi_2} \cup (B_3 \cap x_3^{\phi_3}) \cup (B_2 \cap B_3) \cup (B_2 \cap x_3^{\Diamond\psi_b}) \cup$$
$$(B_1 \cap B_2) \cup (B_1 \cap B_3) \cup (B_1 \cap x_3^{\Diamond\psi_b})$$

We can compute

$$A_\emptyset := (B_2 \cap B_3) \cup (B_1 \cap B_2) \cup (B_1 \cap B_3) = B_3$$

as the coefficient of $\emptyset \subseteq \mathsf{Var}_3^{\phi_{\mathsf{ex}}}$ in the normal form of $\Phi_3(\phi_1)$. Similarly, we obtain $A_{\{x_3^{\phi_2}\}} := \Delta^{\mathsf{N_V}}$, $A_{\{x_3^{\phi_3}\}} := B_3$, and $A_{\{x_3^{\Diamond\psi_b}\}} := B_1 \cup B_2 = B_1$. All other sets of answers A_X are empty.

However, in general we need to consider all eight subsets of $\mathsf{Var}_3^{\phi_{\mathsf{ex}}}$ in such a normal form, which is similar to the number of auxiliary relations needed in Section 5.3 for the formula $\phi_{f''_{\phi_{\mathsf{ex}}}}$ (determined by the number of past operators). In this example, the space requirements of the two approaches do not differ much since the number of alternations between past and future operators in ϕ_{ex} is small.

We can now summarize our history encoding for ϕ as follows. The set of encodings $\Delta^{\mathcal{E}}$ consists of all functions of the form $\Phi\colon \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i$ that are i-bounded, together with a distinct element $I^{\mathcal{E}}$ that marks the beginning of the monitoring process. The transition function $\delta^{\mathcal{E}}$ computes, on input Φ and $\mathcal{I}_i$, the function Φ_i as detailed in Section 6.1, and then transforms all its answer terms into normal form. Finally, the evaluation function $\phi^{\mathcal{E}}$ is given by $\phi^{\mathcal{E}}(\Phi) := \mathsf{eval}^n(\Phi(\phi))$, where the time point n is identified by the (unique) index of the variables in $\Phi(\phi)$. If this answer term contains no variables, then n is irrelevant since eval^n amounts to a simple computation of unions and intersections of sets of answers.

Lemma 6.9. *The history encoding $(\Delta^{\mathcal{E}}, I^{\mathcal{E}}, \delta^{\mathcal{E}}, \phi^{\mathcal{E}})$ for ϕ is correct and bounded.*

Proof. By Lemmata 6.3 and 6.5 and Proposition 6.8, we obtain

$$\begin{aligned}\phi^{\mathcal{E}}(\mathcal{E}(\mathfrak{I}^{(n)})) &= \mathsf{eval}^n\big(\mathcal{E}(\mathfrak{I}^{(n)})(\phi)\big)\\ &= \mathsf{eval}^n(\Phi_n(\phi)) = \mathsf{Ans}(\phi, \mathfrak{I}^{(n)}).\end{aligned}$$

Furthermore, since $\mathcal{E}(\mathfrak{I}^{(n)})$ always contains only answer terms that are in normal form, its size is bounded by $|\mathsf{Sub}(\phi)| \cdot 2^{|\mathsf{FSub}(\phi)|} \cdot |\Delta^{\mathsf{N_V}}|$, which is independent of n. □

The factor $|\Delta^{\mathsf{N_V}}|$ arises from the fact that we always deal with fully evaluated sets of answers. This cannot be avoided since the temporal database monitoring problem anyway requires to compute the sets $\mathsf{Ans}(\phi, \mathfrak{I}^{(n)}) \subseteq \Delta^{\mathsf{N_V}}$.

We now analyze the overall time and space requirements of our approach. For this, let $s, t\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be two functions such that, given a finite interpretation $\mathcal{I}$ over the domain Δ and an *atemporal* query ψ, we can compute $\mathsf{Ans}(\psi, \mathcal{I})$ in time at most $t(|\psi|, |\Delta|)$ and space at most $s(|\psi|, |\Delta|)$. Note that these functions are at least exponential in the number of variables in ψ since $\mathsf{Ans}(\psi, \mathcal{I})$ may contain all possible answer tuples. The proof of the following lemma can be found in Appendix C.

Lemma 6.10. *There is a function $f\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ that is exponential in the first component and polynomial in the second such that we can compute each set $\mathsf{Ans}(\phi, \mathfrak{I}^{(n)})$, $n \geq 0$, in time at most $f(|\phi|, |\Delta|) + |\phi| \cdot t(|\phi|, |\Delta|)$ and space at most $f(|\phi|, |\Delta|) + s(|\phi|, |\Delta|)$.*

This means that we can solve the temporal database monitoring problem in exponential time (and space), in addition to whatever resources we need to answer atemporal queries. The size of the data domain Δ, however, contributes only polynomially to the complexity. Furthermore, the exponential factor of $|\Delta^{\mathsf{N_V}}|$ cannot be avoided.

Consider, for example, a temporal CQ over a temporal *DL-Lite*$_{\mathcal{R}}$-knowledge base with fact base $\mathcal{A}_i$. By [26], every atemporal CQ ψ can be rewritten into a UCQ for which it suffices to evaluate it over $\mathcal{A}_i$ viewed under the closed world assumption, which means that $\Delta_{\mathcal{T}} = \mathsf{N_C}$. The rewritten query is of size exponential in the size of ψ and polynomial in the size of $\mathcal{T}$, and thus one can compute $\mathsf{Cert}(\psi, \langle \mathcal{A}_i, \mathcal{T} \rangle)$ in time exponential in the size of ψ and polynomial in the size of $\mathcal{K}$. Note that this runtime already contains a factor of $|\mathsf{N_C}^{\mathsf{FVar}(\psi)}| \sim |\Delta_{\mathcal{T}}^{\mathsf{N_V}}|$, because all answer tuples have to be enumerated. Thus, answering the temporal CQ only adds another exponential factor in the size of the query (the number of future operators) to the total effort required to solve the temporal database monitoring problem. Since we have constructed a bounded history encoding, this effort is the same regardless of the current time point.

While we can use more efficient rewriting approaches (e.g., [38]), we still need at least exponential time in the number of variables to evaluate the atemporal queries. Furthermore, the additional effort required by the temporal operators is completely independent of this.

7. Rigid Unary Predicates in UCQs

We now extend our temporal semantics by designating certain predicates as being *rigid*, which means that their interpretation is not allowed to change over time. When considering only databases, i.e., finite interpretations, such predicates can be expressed by database tables without explicit time stamps or periods, with the intention that the contained information is valid at every time point.

For now, we consider only rigid *unary* predicates. For example, the unary predicate Server 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 [10, 53].

We assume in this section that there is a set $\mathsf{N_{RP}} \subseteq \mathsf{N_P^1}$ of *rigid unary predicates.* 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 3.1 and additionally *respects* the rigid predicates, i.e., it satisfies $P^{\mathcal{I}_i} = P^{\mathcal{I}_j}$ for every $P \in \mathsf{N_{RP}}$ and all indices i, j between 0 and n.

In this section, we present an approach to deal with these predicates under two restrictions. First, we consider only the source query language $\mathcal{Q}_1$ of unions (disjunctions) of *rooted CQs* (see Example 2.7). Recall that all but one of the rewriting results in Example 2.11 considered only UCQs or sublanguages (CQs or instance queries). Since these UCQs are embedded in a temporal query that allows disjunction, we can without loss of generality assume that we are dealing only with CQs. We call temporal queries over this query language *temporal conjunctive queries (TCQs).* The second restriction is that the logic $\mathcal{L}$ must satisfy the additional property that the class of models of a given knowledge base is closed under countable disjoint unions.

We now describe these restrictions in more detail.

7.1. Rooted Conjunctive Queries

Recall from Example 2.7 that CQs are of the form $\phi = \exists y_1, \ldots, y_m.\psi$, where ψ is a finite conjunction of instance queries, which are called the *atoms* of ϕ. We denote the set of all constants occurring in ϕ by $\mathsf{Const}(\phi)$, and similarly the variables by $\mathsf{Var}(\phi)$ and the free variables by $\mathsf{FVar}(\phi)$. Given a variable assignment $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \mathsf{N_C}$, we denote by $\mathfrak{a}(\phi)$ the Boolean CQ resulting from replacing all free variables in ϕ according to $\mathfrak{a}$.

For the definition of the query language, we also need to define the satisfaction relation $\models$ (cf. Definition 2.5). As usual, it is given using the notion of a homomorphism [54].

Definition 7.1 (semantics of CQs). Let ϕ be a CQ, $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ an interpretation, and $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \mathsf{N_C}$ a variable assignment. A mapping $\pi\colon \mathsf{Var}(\phi) \cup \mathsf{N_C} \to \Delta$ is a *homomorphism* of ϕ into $\mathcal{I}$ (w.r.t. $\mathfrak{a}$) if

- $\pi(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{N_C}$, and
- $(\pi(z_1), \ldots, \pi(z_n)) \in P^{\mathcal{I}}$ for all atoms $P(z_1, \ldots, z_n)$ in $\mathfrak{a}(\phi)$.

We define the satisfaction relation $\models$ by setting $\mathcal{I} \models \mathfrak{a}(\phi)$ iff there is a homomorphism of ϕ into $\mathcal{I}$ w.r.t. $\mathfrak{a}$.

It is a simple matter to check that this query language satisfies the conditions of Definition 2.5.

Intuitively, rooted CQs [41, 55] are CQs that refer to at least one constant (either directly or via a free variable); that is, they are rooted in the named part of an interpretation.

Definition 7.2. A CQ ϕ is called *rooted* if

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

(ii) it is *connected*, i.e., for all $x, y \in \mathsf{Var}(\phi) \cup \mathsf{Const}(\phi)$ there is a sequence $x_1, \dots, x_n \in \mathsf{Var}(\phi) \cup \mathsf{Const}(\phi)$ such that $x_1 = x$, $x_n = y$, and for all i, $1 \leq i \leq n$, there is an atom of ϕ that contains both x_i and x_{i+1}.

A TCQ is *rooted* if it contains only rooted CQs.

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. Note that Condition (ii) on its own does not impose a restriction since any CQ that is not connected can simply be replaced by the conjunction of CQs representing its maximal connected subsets of atoms [29, 56].

To specify the second restriction, we first need to define the countable disjoint union of interpretations.

Definition 7.3. Let $(\mathcal{I}_i)_{i\in I}$ be a countable family of interpretations $\mathcal{I}_i = (\Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i})$ with disjoint domains. For some distinguished $j \in I$, the *disjoint union* of this family (with *core* $\mathcal{I}_j$) is the interpretation $\mathcal{J}$ over the domain $\Delta^{\mathcal{J}} := \bigcup_{i\in I} \Delta^{\mathcal{I}_i}$ with

- $c^{\mathcal{J}} := c^{\mathcal{I}_j}$ for all $c \in \mathsf{N_C}$, and
- $P^{\mathcal{J}} := \bigcup_{i\in I} P^{\mathcal{I}_i}$ for all $n \geq 0$ and $P \in \mathsf{N}_\mathsf{P}^n$.

In the following, we assume that the class of models of any knowledge base in $\mathcal{L}$ is closed under taking disjoint unions. In particular, this means that $\mathcal{L}$-theories are not allowed to place global restrictions on the number of domain elements (of a particular type).

7.2. Rewriting with Rigid Unary Predicates

Before we reconsider the temporal database monitoring problem, we have to verify that Theorem 4.1 remains valid under the new semantics. The main problem we have to solve is that the sequence $\mathfrak{I}_{\mathcal{K}}$ of canonical models does not necessarily respect the rigid predicates. In the following, we make the same assumptions as in Section 4, but for the special case that $\mathcal{Q}_1$ contains only rooted CQs and models of knowledge bases in $\mathcal{L}$ are closed under disjoint unions. For technical reasons, we also assume that $\mathcal{Q}_1$ contains at least all unary instance queries; again, this is satisfied by most results listed in Example 2.11.

Let $\mathcal{K} = \langle(\mathcal{A}_i)_{i\geq 0}, \mathcal{T}\rangle$ be an infinite TKB that represents the sensor data from our system. As usual, we assume that this data is *consistent*, which in this setting means that there is an infinite sequence $\mathfrak{I} = (\mathcal{I}_i)_{i\geq 0}$ of interpretations that respect the rigid predicates such that $\mathcal{I}_i \models \mathcal{A}_i$ and $\mathcal{I}_i \models \mathcal{T}$ for all $i \geq 0$. The finite prefixes $\mathcal{K}^{(n)}$ are then also consistent. We show how to construct modified sequences of interpretations (similar to $\mathfrak{I}_{\mathcal{K}^{(n)}}$ from Theorem 4.1) that respect rigid predicates.

The first step is to find a set of assertions

$$\mathcal{R} \subseteq \{P(c) \mid P \in \mathsf{N_{RP}},\ c \in \mathsf{N_C}\}$$

that specifies the rigid predicates that the constants are allowed to satisfy. Note that $\mathcal{R}$ is always finite since $\mathsf{N_{RP}}$ and $\mathsf{N_C}$ are finite. We denote by $\mathfrak{R}$ the set of all sets of this form. In order to answer TCQs over $\mathcal{K}^{(n)}$, it suffices to consider the TKB $\mathcal{K}_{\mathcal{R}}^{(n)} := \langle(\mathcal{A}_i \cup \mathcal{R})_{0\leq i\leq n}, \mathcal{T}\rangle$ for a suitable $\mathcal{R} \in \mathfrak{R}$. The proof of the following lemma can be found in Appendix D.

Lemma 7.4. *Let $\mathcal{K} = \langle(\mathcal{A}_i)_{i\geq 0}, \mathcal{T}\rangle$ be a consistent infinite TKB. Then there is a set $\mathcal{R} \in \mathfrak{R}$ such that $\mathcal{K}_{\mathcal{R}}^{(n)}$ is consistent for all $n \geq 0$, and for every TCQ ϕ and all i and n with $0 \leq i \leq n$, we have*

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

Given such a set $\mathcal{R}$, we can now construct a sequence of interpretations that respects the rigid predicates and allows us to prove Theorem 4.1 under our new semantics. The details of this construction can also be found in Appendix D.

Theorem 7.5. *Let $\mathcal{Q}_1$, $\mathcal{Q}_2$ be query languages such that $\mathcal{Q}_1$ contains only rooted CQs and $\mathcal{L}$ be a logic that has the canonical model property w.r.t. $\mathcal{Q}_1$ such that $\mathcal{Q}_1$-queries are $\mathcal{Q}_2$-rewritable w.r.t. $\mathcal{L}$. Let further $\mathcal{K} = \langle(\mathcal{A}_i)_{i\geq 0}, \mathcal{T}\rangle$ be a consistent infinite TKB and $\mathcal{R}$ given by Lemma 7.4.*

Then for all $n \geq 0$ there is a sequence of interpretations $\mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}} = (\mathcal{J}_i)_{0\leq i\leq n}$ such that for every temporal $\mathcal{Q}_1$-query ϕ, and all i, $0 \leq i \leq n$, we have

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

Thus, we have again arrived at the temporal database monitoring problem. In contrast to Theorem 4.1, however, we also have to find a suitable set $\mathcal{R}$ in order to obtain the finite interpretations $\mathcal{D}_{\langle\mathcal{A}_i\cup\mathcal{R},\mathcal{T}\rangle}$.

7.3. A Modified History Encoding

Since at the beginning of the monitoring process we have no information except for that given by the background theory about the rigid predicates, we have to consider all sets $\mathcal{R} \in \mathfrak{R}$ as candidates to compute $\mathcal{D}_{\langle\mathcal{A}_0\cup\mathcal{R},\mathcal{T}\rangle}$. But we do not only have to compute $2^{|\mathsf{N_{RP}}|\cdot|\mathsf{N_C}|}$ many rewritten interpretations, the effort required to answer the temporal $\mathcal{Q}_1$-query ϕ is also increased by the same factor.

However, one can clearly eliminate those $\mathcal{R} \in \mathfrak{R}$ from consideration for which we find out that $\langle\mathcal{A}_i \cup \mathcal{R}, \mathcal{T}\rangle$ is inconsistent at some point. Since we have assumed that the sensor and background data is consistent, this can only happen if the set $\mathcal{R}$ is the wrong set, i.e., not the one whose existence is guaranteed by Lemma 7.4.

More formally, let ϕ be a temporal $\mathcal{Q}_1$-query and $(\Delta^{\mathcal{E}}, I^{\mathcal{E}}, \delta^{\mathcal{E}}, \phi^{\mathcal{E}})$ be a history encoding for $\phi^{\mathcal{T}}$ (without rigid predicates). We describe an algorithm to deal with rigid unary predicates that is similar to a history encoding, but directly reads the fact bases $\mathcal{A}_i$ instead of the interpretations $\mathcal{D}_{\langle \mathcal{A}_i \cup \mathcal{R}, \mathcal{T}\rangle}$. Recall that the latter are finite interpretations over the domain $\Delta_{\mathcal{T}}$, which we again assume to be equal to $\mathsf{N_C}$ in the following.

Algorithm 7.6. The main data structure is a partial function $f\colon \mathfrak{R} \to \Delta^{\mathcal{E}}$ that specifies encodings corresponding to some of the sets $\mathcal{R} \in \mathfrak{R}$. On input $\mathcal{T}$, the algorithm does the following:

1. For $\mathcal{R} \in \mathfrak{R}$, the value $f(\mathcal{R})$ is initialized to $I^{\mathcal{E}}$ whenever $\langle \mathcal{R}, \mathcal{T}\rangle$ is consistent, and remains undefined otherwise.

2. Let f contain the current encodings and $\mathcal{A}$ be the next fact base. For every $\mathcal{R} \in \mathfrak{R}$, the new encoding is obtained as $f'(\mathcal{R}) := \delta^{\mathcal{E}}(f(\mathcal{R}), \mathcal{D}_{\langle \mathcal{A}\cup\mathcal{R}, \mathcal{T}\rangle})$ if $f(\mathcal{R})$ is defined and $\langle \mathcal{A} \cup \mathcal{R}, \mathcal{T}\rangle$ is consistent. All other values $f'(\mathcal{R})$ remain undefined.

3. The current encodings are now given by $f := f'$ and the value
$$\bigcap_{\substack{\mathcal{R}\in\mathfrak{R}\\ f(\mathcal{R}) \text{ is defined}}} \phi^{\mathcal{E}}(f(\mathcal{R}))$$
is returned. Continue with Step 2.

Intuitively, we run several instances of the original history encoding in parallel, with the only difference between them being that each instance has a different 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 input fact bases, the corresponding instance is stopped. Each remaining instance computes the certain answers to ϕ relative to one set $\mathcal{R}$, and the actual set of certain answers to ϕ is then computed as their intersection.

The consistency tests for $\langle \mathcal{A} \cup \mathcal{R}, \mathcal{T}\rangle$ are necessary since $\mathcal{D}_{\langle \mathcal{A}\cup\mathcal{R},\mathcal{T}\rangle}$ is only defined if the knowledge base is consistent. Furthermore, this allows us to remove sets $\mathcal{R}$ from consideration, which makes the algorithm more efficient. The hope is that, over time, more and more sets $\mathcal{R}$ are discarded because of new information until only few of them remain.

We show that this computation preserves correctness and boundedness of the given history encoding in a sense similar to that of Definition 5.1.

Theorem 7.7. *Let ϕ be a temporal $\mathcal{Q}_1$-query. Given a correct history encoding for $\phi^{\mathcal{T}}$ and a consistent infinite TKB $\mathcal{K} = \langle (\mathcal{A}_i)_{i\geq 0}, \mathcal{T}\rangle$, Algorithm 7.6 outputs* $\mathsf{Cert}(\phi, \mathcal{K}^{(n)})$ *for each $n \geq 0$. If the history encoding is bounded, then the size of f does not depend on n.*

Proof. The second claim holds since the size of f is at most $2^{|\mathsf{N_{RP}}|\cdot|\mathsf{N_C}|}$ times the size of $\mathcal{E}(\mathfrak{D}_{\mathcal{K}^{(n)}_{\mathcal{R}}})$, and both values are independent of n.

For the correctness, observe first that for every $\mathcal{R}$ for which $f(\mathcal{R})$ is defined at time point $n \geq 0$, we have
$$\phi^{\mathcal{E}}(f(\mathcal{R})) = \mathsf{Ans}(\phi^{\mathcal{T}}, \mathfrak{D}_{\mathcal{K}^{(n)}_{\mathcal{R}}}) = \mathsf{Cert}(\phi, \mathcal{K}^{(n)}_{\mathcal{R}})$$
by Definition 5.1 and Theorem 7.5. This set always contains $\mathsf{Cert}(\phi, \mathcal{K}^{(n)})$ since $\mathcal{K}^{(n)}_{\mathcal{R}}$ is more restrictive than $\mathcal{K}^{(n)}$. Furthermore, by Lemma 7.4, we know that there must be at least one $\mathcal{R} \in \mathfrak{R}$ that passes all consistency test such that $\mathsf{Cert}(\phi, \mathcal{K}^{(n)}_{\mathcal{R}})$ is even equal to $\mathsf{Cert}(\phi, \mathcal{K}^{(n)})$. This shows that the intersection in Step 3 yields $\mathsf{Cert}(\phi, \mathcal{K}^{(n)})$. □

Thus, every correct history encoding can be extended to deal with rigid unary predicates while increasing the time and space requirements by a factor of $2^{|\mathsf{N_{RP}}|\cdot|\mathsf{N_C}|}$. For the bounded history encoding of Section 6, this means that its total resource consumption at each time point is proportional to $|\mathsf{Sub}(\phi)| \cdot 2^{|\mathsf{FSub}(\phi)|} \cdot |\Delta_{\mathcal{T}}^{\mathsf{FVar}(\phi)}| \cdot 2^{|\mathsf{N_{RP}}|\cdot|\mathsf{N_C}|}$ plus $2^{|\mathsf{N_{RP}}|\cdot|\mathsf{N_C}|}$ times the requirements for answering the rewritten atemporal $\mathcal{Q}_2$-subqueries over finite interpretations over the domain $\Delta_{\mathcal{T}}$ (cf. Lemma 6.10).

8. Discussion

In this article, we have introduced a generic temporal query language that combines the well-known temporal logic LTL with queries over knowledge bases. Further, we have shown how the reasoning task of *temporal OBDA* over knowledge bases is reduced to answering queries over temporal databases, similar to what was done for the atemporal case (see Example 2.11). We then presented three approaches that solve the resulting *temporal database monitoring problem* and described an approach to extend any history encoding to deal with rigid unary predicates for the special case where only rooted conjunctive queries are allowed. In what follows, we describe advantages and drawbacks of the former three approaches.

8.1. Comparison

We focus on the required implementation effort, on aspects of the implementation, as well as on the amount of memory required. We thus point out characteristics that can guide the choice of a particular approach for a specific use case.

First approach. The most straightforward option is to evaluate TQs in a database system that supports dealing with temporal information using a suitable translation (see Section 5.1). The advantage of this is that one can directly exploit database optimization techniques. However, it requires storing the whole history of past sensor data (even if only a small part of it is necessary to answer the query) and re-evaluating the query at each time point using a temporal database query language like ATSQL [15]. As the length of the history can get very long, this is not the preferred option. Nevertheless, this approach may still be feasible if the amount of data can be limited by other means, such as adopting a "sliding view" semantics where only a fixed amount of past time points is used to evaluate temporal queries.

Second approach. The approach described in Section 5.3 is based on the bounded history encoding from [16, 18]. Any implementation of this approach has to eliminate the future operators in the query; we described how this elimination can be done. Although independent of the length of the history, this step involves a theoretical non-elementary blowup in the size of the query due to the use of the separation theorem [52]. Even for propositional LTL, this translation is at least exponential and no approach less than triply exponential is known [5]. An advantage of the history encoding from [16, 18] is that it can be implemented inside a database system using views and triggers, which could yield a good performance in spite of the possibly very large size of the query. Generally, this option is the best of the three if the TQ contains no future operators or if one can find a small equivalent representation without future operators.

Third approach. The most general solution is based on the answer terms described in Section 6. The presented algorithm is an adaptation of the one in [16] and works directly with future operators by introducing place-holder variables for future answers. We have shown that this also achieves a bounded history encoding while we can limit the influence of the future operators on the time and space requirements to a single exponential factor. However, it is not straightforward how to implement this approach inside a database system. For that it remains to be investigated how the implementation inside a database system described in [16] can be extended to cover answer terms in an efficient way, in particular in the presence of the place-holder variables. Even using the normal form described in Definition 6.7, we still need to store exponentially many sets of answer tuples, and it is not clear whether they can be accessed through views. While theoretically the most efficient solution, it remains to be seen how it performs in practice in an optimized implementation.

8.2. Outlook

In future work, we want to implement our proposed algorithm, and compare the performance of all three described approaches on realistic queries over temporal relational databases to see which approach best suits context-aware applications. In particular, it is likely that the approach from Section 5.3 outperforms the dedicated algorithm from Section 6 on certain kinds of TQs, e.g., queries with a small bound on the nesting depth of the temporal operators.

On the theoretical side, we plan to investigate how to adapt the algorithm to deal also with rigid n-ary predicates for $n > 1$. It would also be interesting to find out whether one can extend the bounded history encoding from Section 6 to deal with negation in the query language if queries are assumed to be domain-independent, which is already possible with the approaches in [15, 16].

Acknowledgments

This work was partially supported by the DFG in the Collaborative Research Center 912 (HAEC) and in the Research Training Group 1763 (QuantLA). We also thank Franz Baader for helpful discussions on the topics of temporal logics and monitoring, and the anonymous reviewers for their suggestions for improving the paper.

[1] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.

[2] S. Decker, M. Erdmann, D. Fensel, R. Studer, Ontobroker: Ontology based access to distributed and semi-structured information, in: R. Meersman, Z. Tari, S. M. Stevens (Eds.), Proceedings of the 8th Working Conference on Database Semantics (DS-8), Vol. 138 of IFIP Conference Proceedings, Kluwer, 1999, pp. 351–369.

[3] A. Poggi, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Linking data to ontologies, Journal on Data Semantics X (2008) 133–173.

[4] A. Pnueli, The temporal logic of programs, in: Proc. of the 18th Annual Symp. on Foundations of Computer Science (FOCS'77), IEEE Press, 1977, pp. 46–57.

[5] F. Laroussinie, N. Markey, P. Schnoebelen, Temporal logic with forgettable past, in: Proc. of the 17th Annual IEEE Symp. on Logic in Computer Science (LICS'02), IEEE Press, 2002, pp. 383–392.

[6] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, 2nd Edition, Cambridge University Press, 2007.

[7] A. Artale, E. Franconi, F. Wolter, M. Zakharyaschev, A temporal description logic for reasoning over conceptual schemas and queries, in: Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA 2002), Springer-Verlag, 2002, pp. 98–110.

[8] B. Motik, Representing and querying validity time in RDF and OWL: A logic-based approach, Journal of Web Semantics 12–13 (2012) 3–21.

[9] V. Gutiérrez-Basulto, S. Klarman, Towards a unifying approach to representing and querying temporal data in description logics, in: M. Krötzsch, U. Straccia (Eds.), Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems (RR'12), Vol. 7497 of Lecture Notes in Computer Science, Springer-Verlag, 2012, pp. 90–105.

[10] F. Baader, S. Borgwardt, M. Lippmann, Temporalizing ontology-based data access, in: M. P. Bonacina (Ed.), Proc. of the 24th Int. Conf. on Automated Deduction (CADE'13), Vol. 7898 of Lecture Notes in Artificial Intelligence, Springer-Verlag, 2013, pp. 330–344.

[11] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, Ontologies and databases: The DL-Lite approach, in: S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh, M.-C. Rousset, R. A. Schmidt (Eds.), Reasoning Web, 5th Int. Summer School 2009, Tutorial Lectures, Vol. 5689 of Lecture Notes in Computer Science, Springer-Verlag, 2009, pp. 255–356.

[12] C. Lutz, F. Wolter, M. Zakharyaschev, Temporal description logics: A survey, in: S. Demri, C. S. Jensen (Eds.), Proceedings of the 15th International Symposium on Temporal Representation and Reasoning (TIME 2008), IEEE Press, 2008, pp. 3–14.

[13] A. Artale, R. Kontchakov, C. Lutz, F. Wolter, M. Zakharyaschev, Temporalising tractable description logics, in: V. Goranko, X. S.

Wang (Eds.), Proceedings of the 14th International Symposium on Temporal Representation and Reasoning (TIME 2007), IEEE Press, 2007, pp. 11–22.

[14] A. Artale, R. Kontchakov, F. Wolter, M. Zakharyaschev, Temporal description logic for ontology-based data access, in: F. Rossi (Ed.), Proceedings of the 23rd Interntational Joint Conference on Artificial Intelligence (IJCAI 2013), AAAI Press, 2013, pp. 711–717.

[15] J. Chomicki, D. Toman, M. H. Böhlen, Querying ATSQL databases with temporal logic, ACM Transactions on Database Systems 26 (2) (2001) 145–178.

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

[17] J. Chomicki, D. Toman, Time in database systems, in: M. Fisher, D. Gabbay, Lluis Vila (Eds.), Handbook of Temporal Reasoning in Artificial Intelligence, Elsevier, 2005, pp. 429–467.

[18] D. Toman, Logical data expiration, in: J. Chomicki, R. van der Meyden, G. Saake (Eds.), Logics for Emerging Applications of Databases, Springer-Verlag, 2004, Ch. 6, pp. 203–238.

[19] S. Borgwardt, M. Lippmann, V. Thost, Temporal query answering in the description logic *DL-Lite*, in: P. Fontaine, C. Ringeissen, R. A. Schmidt (Eds.), Proceedings of the 9th International Symposium on Frontiers of Combining Systems (FroCoS 2013), Vol. 8152 of Lecture Notes in Computer Science, Springer-Verlag, 2013, pp. 165–180.

[20] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, *DL-Lite*: Tractable description logics for ontologies, in: M. M. Veloso, S. Kambhampati (Eds.), Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI'05), AAAI Press, 2005, pp. 602–607.

[21] F. Baader, Terminological cycles in a description logic with existential restrictions, in: G. Gottlob, T. Walsh (Eds.), Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03), Morgan Kaufmann, 2003, pp. 325–330.

[22] I. Horrocks, O. Kutz, U. Sattler, The even more irresistible $\mathcal{SROIQ}$, in: P. Doherty, J. Mylopoulos, C. Welty (Eds.), Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06), AAAI Press, 2006, pp. 57–67.

[23] U. Hustadt, B. Motik, U. Sattler, Reasoning in description logics by a reduction to disjunctive datalog, Journal of Automated Reasoning 39 (3) (2007) 351–384.

[24] Y. Kazakov, Consequence-driven reasoning for horn $\mathcal{SHIQ}$ ontologies, in: C. Boutilier (Ed.), Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09), AAAI Press, 2009, pp. 2040–2045.

[25] A. Calì, G. Gottlob, T. Lukasiewicz, A general Datalog-based framework for tractable query answering over ontologies 14 (2012) 57–83.

[26] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family, Journal of Automated Reasoning 39 (3) (2007) 385–429.

[27] B. Glimm, I. Horrocks, C. Lutz, U. Sattler, Conjunctive query answering for the description logic $\mathcal{SHIQ}$, Journal of Artificial Intelligence Research 31 (1) (2008) 157–204.

[28] C. Lutz, D. Toman, F. Wolter, Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system, in: C. Boutilier (Ed.), Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09), AAAI Press, 2009, pp. 2070–2075.

[29] S. Rudolph, B. Glimm, Nominals, inverses, counting, and conjunctive queries or: Why infinity is your friend!, Journal of Artificial Intelligence Research 39 (1) (2010) 429–481.

[30] M. Bienvenu, B. ten Cate, C. Lutz, F. Wolter, Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP, in: R. Hull, W. Fan (Eds.), Proc. of the 32nd Symp. on Principles of Database Systems (PODS'13), ACM, 2013, pp. 213–224.

[31] S. Abiteboul, V. Vianu, Regular path queries with constraints, Journal of Computer and System Sciences 58 (3) (1999) 428–452.

[32] D. Calvanese, G. De Giacomo, M. Lenzerini, M. Y. Vardi, Rewriting of regular expressions and regular path queries, Journal of Computer and System Sciences 64 (3) (2002) 443–465.

[33] H. Pérez-Urbina, B. Motik, I. Horrocks, Tractable query answering and rewriting under description logic constraints, Journal of Applied Logic 8 (2) (2010) 186–209.

[34] T. Eiter, M. Ortiz, M. Šimkus, T.-K. Tran, G. Xiao, Query rewriting for horn-$\mathcal{SHIQ}$ plus rules, in: J. Hoffmann, B. Selman (Eds.), Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI'12), AAAI Press, 2012, pp. 726–733.

[35] F. Baader, S. Brandt, C. Lutz, Pushing the $\mathcal{EL}$ envelope., in: L. P. Kaelbling, A. Saffiotti (Eds.), Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05), Professional Book Center, 2005, pp. 364–369.

[36] R. Rosati, On conjunctive query answering in $\mathcal{EL}$, in: D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, A.-Y. Turhan, S. Tessaris (Eds.), Proc. of the 2007 Int. Workshop on Description Logics (DL'07), Vol. 250 of CEUR Workshop Proceedings, 2007, pp. 451–458.

[37] A. Krisnadhi, C. Lutz, Data complexity in the $\mathcal{EL}$ family of description logics, in: N. Dershowitz, A. Voronkov (Eds.), Proc. of the 14th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07), Vol. 4790 of Lecture Notes in Computer Science, Springer-Verlag, 2007, pp. 333–347.

[38] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, M. Zakharyaschev, The combined approach to query answering in *DL-Lite*, in: F. Lin, U. Sattler, M. Truszczynski (Eds.), Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'10), AAAI Press, 2010, pp. 247–257.

[39] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, M. Zakharyaschev, The combined approach to ontology-based data access, in: T. Walsh (Ed.), Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11), AAAI Press, 2011, pp. 2656–2661.

[40] M. Ortiz, S. Rudolph, M. Šimkus, Query answering in the horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$, in: T. Walsh (Ed.), Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11), AAAI Press, 2011, pp. 1039–1044.

[41] M. Bienvenu, M. Ortiz, M. Šimkus, G. Xiao, Tractable queries for lightweight description logics, in: F. Rossi (Ed.), Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI'13), AAAI Press, 2013, pp. 768–774.

[42] G. Gottlob, G. Orsi, A. Pieris, Ontological queries: Rewriting and optimization, in: Proc. of the 2011 IEEE 27th Int. Conf. on Data Engineering (ICDE'11), IEEE Computer Society Press, 2011, pp. 2–13.

[43] R. Rosati, A. Almatelli, Improving query answering over *DL-Lite* ontologies, in: F. Lin, U. Sattler, M. Truszczynski (Eds.), Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'10), AAAI Press, 2010, pp. 290–300.

[44] S. Heymans, T. Eiter, G. Xiao, Tractable reasoning with dl-programs over Datalog-rewritable description logics, in: H. Coelho, R. Studer, M. Wooldridge (Eds.), Proc. of the 19th Eur. Conf. on Artificial Intelligence (ECAI'10), Vol. 215 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2010, pp. 35–40.

[45] M. Krötzsch, Efficient rule-based inferencing for OWL EL, in: T. Walsh (Ed.), Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11), AAAI Press, 2011, pp. 2668–2773.

[46] T. Eiter, Datalog-based data access over ontology knowledge bases, in: Semantic Web - Ontology Languages and Their Use. ICCL Summer School, Tutorial Lectures, 2013.

[47] M. Bienvenu, C. Lutz, F. Wolter, First-order rewritability of atomic queries in horn description logics, in: F. Rossi (Ed.), Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI'13), AAAI Press, 2013, pp. 754–760.

[48] T. Wilke, Classifying discrete temporal properties, in: Proc. of the 16th Annual Symp. on Theoretical Aspects of Computer Science (STACS'99), Vol. 1563 of Lecture Notes in Computer Science, Springer-Verlag, 1999, pp. 32–46.

[49] K. Hülsmann, G. Saake, Theoretical foundations of handling large substitution sets in temporal integrity monitoring, Acta

Informatica 28 (4) (1991) 365–407.

[50] G. Saake, U. W. Lipeck, Using finite-linear temporal logic for specifying database dynamics, in: E. Börger, H. K. Büning, M. M. Richter (Eds.), Proc. of the 2nd Workshop on Computer Science Logic (CSL'88), Vol. 385 of Lecture Notes in Computer Science, Springer-Verlag, 1989, pp. 288–300.

[51] M. H. Böhlen, C. S. Jensen, R. T. Snodgrass, Temporal statement modifiers, ACM Transactions on Database Systems 25 (4) (2000) 407–456.

[52] D. Gabbay, Declarative past and imperative future, in: B. Banieqbal, H. Barringer, A. Pnueli (Eds.), Proc. of the 1987 Coll. on Temporal Logic in Specification, Vol. 398 of Lecture Notes in Computer Science, Springer-Verlag, 1989, pp. 409–448.

[53] F. Baader, S. Ghilardi, C. Lutz, LTL over description logic axioms, ACM Transactions on Computational Logic 13 (3) (2012) 21:1–21:32.

[54] A. K. Chandra, P. M. Merlin, Optimal implementation of conjunctive queries in relational data bases, in: J. E. Hopcroft, E. P. Friedman, M. A. Harrison (Eds.), Proc. of the 9th Annual ACM Symp. on Theory of Computing (STOC'77), ACM Press, 1977, pp. 77–90.

[55] C. Lutz, The complexity of conjunctive query answering in expressive description logics, in: Proc. of the 4th Int. Joint Conf. on Automated Reasoning (IJCAR'08), Vol. 5195 of Lecture Notes in Artificial Intelligence, Springer-Verlag, 2008, pp. 179–193.

[56] S. Tessaris, Questions and answers: Reasoning and querying in description logic, Ph.D. thesis, University of Manchester (2001).

Appendix A. Proof of Theorem 4.1

Let $\mathcal{Q}_1$, $\mathcal{Q}_2$ be query languages and $\mathcal{L}$ be a logic that has the canonical model property w.r.t. $\mathcal{Q}_1$ such that $\mathcal{Q}_1$-queries are $\mathcal{Q}_2$-rewritable w.r.t. $\mathcal{L}$. Then for every consistent TKB $\mathcal{K} = \langle(\mathcal{A}_i)_{0\leq i\leq n}, \mathcal{T}\rangle$, every temporal $\mathcal{Q}_1$-query ϕ, and every i, $0 \leq i \leq n$, we have

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

Proof. We first prove $\mathsf{Cert}(\phi, \mathcal{K}, i) \subseteq \mathsf{Ans}(\phi, \mathfrak{I}_\mathcal{K}, i)$. Take $\mathfrak{a} \in \mathsf{Cert}(\phi, \mathcal{K}, i)$. Then for every $\mathfrak{I} = (\mathcal{I}_i)_{0\leq i\leq n}$ with $\mathfrak{I} \models \mathcal{K}$, we have $\mathfrak{I}, i \models_{\mathcal{Q}_1} \mathfrak{a}(\phi)$. In particular, we get $\mathfrak{I}_\mathcal{K}, i \models_{\mathcal{Q}_1} \mathfrak{a}(\phi)$, 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) $\mathsf{Ans}(\phi, \mathfrak{I}_\mathcal{K}, i) \subseteq \mathsf{Ans}(\phi^\mathcal{T}, \mathfrak{D}_\mathcal{K}, i)$, and

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

We show this by induction on the structure of ϕ.

For the base case, consider an atemporal $\mathcal{Q}_1$-query ϕ. For (1), take $\mathfrak{a} \in \mathsf{Ans}(\phi, \mathfrak{I}_\mathcal{K}, i)$. Since ϕ is a $\mathcal{Q}_1$-query, the semantics yields that $\mathfrak{a} \in \mathsf{Ans}(\phi, \mathcal{I}_{\mathcal{K}_i})$. By $\mathcal{Q}_2$-rewritability, we obtain $\mathfrak{a} \in \mathsf{Ans}(\phi^\mathcal{T}, \mathcal{D}_{\mathcal{K}_i})$. Finally, the semantics of temporal $\mathcal{Q}_2$-queries yields that $\mathfrak{a} \in \mathsf{Ans}(\phi^\mathcal{T}, \mathfrak{D}_\mathcal{K}, i)$.

For (2), take $\mathfrak{a} \in \mathsf{Ans}(\phi^\mathcal{T}, \mathfrak{D}_\mathcal{K}, i)$. Since $\phi^\mathcal{T}$ is a $\mathcal{Q}_2$-query, this implies that $\mathfrak{a} \in \mathsf{Ans}(\phi^\mathcal{T}, \mathcal{D}_{\mathcal{K}_i})$. Because of $\mathcal{Q}_2$-rewritability, we have $\mathfrak{a} \in \mathsf{Cert}(\phi, \mathcal{K}_i)$. This means that for every interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}_i$ and $\mathcal{I} \models \mathcal{T}$, we have that $\mathcal{I} \models_{\mathcal{Q}_1} \mathfrak{a}(\phi)$. Hence, for every sequence $\mathfrak{I} = (\mathcal{I}_i)_{0\leq i\leq n}$ with $\mathfrak{I} \models \mathcal{K}$, we have $\mathcal{I}_i \models_{\mathcal{Q}_1} \mathfrak{a}(\phi)$. Since ϕ is a $\mathcal{Q}_1$-query, the latter condition is equivalent to $\mathfrak{a} \in \mathsf{Ans}(\phi, \mathfrak{I}, i)$, and thus we get $\mathfrak{a} \in \mathsf{Cert}(\phi, \mathcal{K}, i)$.

Let now ϕ be of the form $\phi_1 \wedge \phi_2$. For (1), assume that $\mathfrak{I}_\mathcal{K}, i \models_{\mathcal{Q}_1} \mathfrak{a}(\phi)$, and thus we have $\mathfrak{I}_\mathcal{K}, i \models_{\mathcal{Q}_1} \mathfrak{a}_{\phi_1}(\phi_1)$ and $\mathfrak{I}_\mathcal{K}, i \models_{\mathcal{Q}_1} \mathfrak{a}_{\phi_2}(\phi_2)$. By the induction hypothesis, $\mathfrak{D}_\mathcal{K}, i \models_{\mathcal{Q}_2} \mathfrak{a}_{\phi_1}(\phi_1^\mathcal{T})$ and $\mathfrak{D}_\mathcal{K}, i \models_{\mathcal{Q}_2} \mathfrak{a}_{\phi_2}(\phi_2^\mathcal{T})$, and thus by the definition of $\phi^\mathcal{T}$ we get $\mathfrak{D}_\mathcal{K}, i \models_{\mathcal{Q}_2} \mathfrak{a}(\phi^\mathcal{T})$.

For (2), assume that $\mathfrak{D}_\mathcal{K}, i \models_{\mathcal{Q}_2} \mathfrak{a}(\phi^\mathcal{T})$, and thus $\mathfrak{D}_\mathcal{K}, i \models_{\mathcal{Q}_2} \mathfrak{a}_{\phi_1}(\phi_1^\mathcal{T})$ and $\mathfrak{D}_\mathcal{K}, i \models_{\mathcal{Q}_2} \mathfrak{a}_{\phi_2}(\phi_2^\mathcal{T})$. 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 $\mathfrak{I}, i \models_{\mathcal{Q}_1} \mathfrak{a}_{\phi_1}(\phi_1)$ and $\mathfrak{I}, i \models_{\mathcal{Q}_1} \mathfrak{a}_{\phi_2}(\phi_2)$. 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 claim (1), we take $\mathfrak{I}_\mathcal{K}, i \models_{\mathcal{Q}_1} \mathfrak{a}(\bigcirc\phi_1)$. By the temporal semantics, we have $i < n$ and $\mathfrak{I}_\mathcal{K}, i+1 \models_{\mathcal{Q}_1} \mathfrak{a}(\phi_1)$. By the induction hypothesis, we get $\mathfrak{D}_\mathcal{K}, i+1 \models_{\mathcal{Q}_2} \mathfrak{a}(\phi_1^\mathcal{T})$. Since $i < n$, this implies that $\mathfrak{D}_\mathcal{K}, i \models_{\mathcal{Q}_2} \mathfrak{a}(\phi^\mathcal{T})$ by the definition of $\phi^\mathcal{T}$.

For (2), let $\mathfrak{D}_\mathcal{K}, i \models_{\mathcal{Q}_2} \mathfrak{a}(\phi^\mathcal{T})$. Hence, we have $i < n$ and $\mathfrak{D}_\mathcal{K}, i+1 \models_{\mathcal{Q}_2} \mathfrak{a}(\phi_1^\mathcal{T})$, 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 $\mathfrak{I}, i \models_{\mathcal{Q}_1} \mathfrak{a}(\bigcirc\phi_1)$, which shows that $\mathfrak{a} \in \mathsf{Cert}(\phi, \mathcal{K}, i)$.

For the next inductive case, let ϕ be of the form $\phi_1 \mathsf{U}\, \phi_2$. For (1), assume that $\mathfrak{I}_\mathcal{K}, i \models_{\mathcal{Q}_1} \mathfrak{a}(\phi_1 \mathsf{U}\, \phi_2)$, and thus there is a k, $i \leq k \leq n$, such that we have $\mathfrak{I}_\mathcal{K}, k \models_{\mathcal{Q}_1} \mathfrak{a}_{\phi_2}(\phi_2)$ and $\mathfrak{I}_\mathcal{K}, j \models_{\mathcal{Q}_1} \mathfrak{a}_{\phi_1}(\phi_1)$ for all j, $i \leq j < k$. By the induction hypothesis, we obtain $\mathfrak{D}_\mathcal{K}, k \models_{\mathcal{Q}_2} \mathfrak{a}_{\phi_2}(\phi_2^\mathcal{T})$ and $\mathfrak{D}_\mathcal{K}, j \models_{\mathcal{Q}_2} \mathfrak{a}_{\phi_1}(\phi_1^\mathcal{T})$ for all j, $i \leq j < k$. The definitions of $\models_{\mathcal{Q}_2}$ and $\phi^\mathcal{T}$ yield that $\mathfrak{D}_\mathcal{K}, i \models_{\mathcal{Q}_2} \mathfrak{a}(\phi^\mathcal{T})$.

For (2), assume that $\mathfrak{D}_\mathcal{K}, i \models_{\mathcal{Q}_2} \mathfrak{a}(\phi^\mathcal{T})$. By definition of $\phi^\mathcal{T}$, there is a k, $i \leq k \leq n$, with $\mathfrak{D}_\mathcal{K}, k \models_{\mathcal{Q}_2} \mathfrak{a}_{\phi_2}(\phi_2^\mathcal{T})$ and $\mathfrak{D}_\mathcal{K}, j \models_{\mathcal{Q}_2} \mathfrak{a}_{\phi_1}(\phi_1^\mathcal{T})$ for all j, $i \leq j < k$. The induction hypothesis yields $\mathfrak{a} \in \mathsf{Cert}(\phi_2, \mathcal{K}, k)$ and $\mathfrak{a} \in \mathsf{Cert}(\phi_1, \mathcal{K}, j)$ for all j, $i \leq j < k$. As a consequence, we have for every $\mathfrak{I} \models \mathcal{K}$ that $\mathfrak{I}, i \models_{\mathcal{Q}_1} \mathfrak{a}(\phi_1 \mathsf{U}\, \phi_2)$.

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 $\mathfrak{I}_\mathcal{K}, i \models_{\mathcal{Q}_1} \mathfrak{a}(\phi)$ and $\mathfrak{D}_\mathcal{K}, i \models_{\mathcal{Q}_2} \mathfrak{a}(\phi^\mathcal{T})$ 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 $\phi_1 \mathsf{U}\, \phi_2$. The cases of $\Box$, $\Box^-$, $\Diamond$, and $\Diamond^-$ follow by similar arguments. □

Appendix B. Reduction of Section 5.3

In this part of the appendix, we describe how to rewrite a TQ ϕ into an equivalent temporal query ϕ' of the language of [16] in order to apply the algorithm described in [16].

We first transform the TQ ϕ into an LTL-formula f_ϕ, which is defined inductively on the structure of ϕ:

ϕ	f_ϕ
$\mathcal{Q}$-query ψ_j	p_j
$\phi_1 \wedge \phi_2$	$f_{\phi_1} \wedge f_{\phi_2}$
$\phi_1 \vee \phi_2$	$f_{\phi_1} \vee f_{\phi_2}$
$\bigcirc\phi_1$	$\mathsf{false}\, \mathsf{U}^{<}(f_{\phi_1} \wedge p)$

$\bigcirc^- \phi_1$	$\mathsf{false}\,\mathsf{S}^< f_{\phi_1}$
$\bullet\phi_1$	$\mathsf{false}\,\mathsf{U}^<(f_{\phi_1} \vee \neg p)$
$\bullet^- \phi_1$	$\mathsf{first} \vee \mathsf{false}\,\mathsf{S}^< f_{\phi_1}$
$\Box\phi_1$	$f_{\phi_1} \wedge f_{\phi_1}\,\mathsf{U}^< \neg p$
$\Box^- \phi_1$	$f_{\phi_1} \wedge f_{\phi_1}\,\mathsf{S}^<(\mathsf{first} \wedge f_{\phi_1})$
$\Diamond\phi_1$	$f_{\phi_1} \vee \mathsf{true}\,\mathsf{U}^<(f_{\phi_1} \wedge p)$
$\Diamond^- \phi_1$	$f_{\phi_1} \vee \mathsf{true}\,\mathsf{S}^< f_{\phi_1}$
$\phi_1\,\mathsf{U}\,\phi_2$	$f_{\phi_2} \vee (f_{\phi_1} \wedge f_{\phi_1}\,\mathsf{U}^<(f_{\phi_2} \wedge p))$
$\phi_1\,\mathsf{S}\,\phi_2$	$f_{\phi_2} \vee (f_{\phi_1} \wedge f_{\phi_1}\,\mathsf{S}^< f_{\phi_2})$

This yields the following lemma, where $\mathfrak{I}_\mathfrak{a}$ is defined as in Section 5.3.

Lemma Appendix B.1. *For all $\mathfrak{I} = (\mathcal{I}_i)_{0\leq i\leq n}$, all $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \mathsf{N_C}$, and all i, $0 \leq i \leq n$, we have $\mathfrak{I}, i \models \mathfrak{a}(\phi)$ iff $\mathfrak{I}_\mathfrak{a}, i \models f_\phi$.*

Proof. We prove this lemma by induction on the structure of ϕ. For the base case of a $\mathcal{Q}$-query $\phi = \psi_j$, we have:

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\psi_j)\\
\textit{iff}\quad &\mathcal{I}_i \models \mathfrak{a}(\psi_j)\\
\textit{iff}\quad &p_j \in w_i\\
\textit{iff}\quad &\mathfrak{I}_\mathfrak{a}, i \models f_{\psi_j}.
\end{aligned}$$

For the case $\phi = \phi_1 \wedge \phi_2$, we have:

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \wedge \phi_2)\\
\textit{iff}\quad &\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ and } \mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2)\\
\textit{iff}\quad &\mathfrak{I}_{\mathfrak{a}_{\phi_1}}, i \models f_{\phi_1} \text{ and } \mathfrak{I}_{\mathfrak{a}_{\phi_2}}, i \models f_{\phi_2}\\
\textit{iff}\quad &\mathfrak{I}_\mathfrak{a}, i \models f_{\phi_1} \text{ and } \mathfrak{I}_\mathfrak{a}, i \models f_{\phi_2}\\
\textit{iff}\quad &\mathfrak{I}_\mathfrak{a}, i \models f_{\phi_1} \wedge f_{\phi_2}.
\end{aligned}$$

For the case $\phi = \bigcirc\phi_1$, we have:

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\bigcirc\phi_1)\\
\textit{iff}\quad &i < n \text{ and } \mathfrak{I}, i+1 \models \mathfrak{a}(\phi_1)\\
\textit{iff}\quad &\mathfrak{I}_\mathfrak{a}, i+1 \models p \text{ and } \mathfrak{I}_\mathfrak{a}, i+1 \models f_{\phi_1}\\
\textit{iff}\quad &\mathfrak{I}_\mathfrak{a}, i+1 \models f_{\phi_1} \wedge p\\
\textit{iff}\quad &\mathfrak{I}_\mathfrak{a}, i \models \mathsf{false}\,\mathsf{U}^<(f_{\phi_1} \wedge p).
\end{aligned}$$

For the case $\phi = \bullet^-\phi_1$, we have:

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\bullet^-\phi_1)\\
\textit{iff}\quad &i > 0 \text{ implies } \mathfrak{I}, i-1 \models \mathfrak{a}(\phi_1)\\
\textit{iff}\quad &i = 0 \text{ or } i > 0 \text{ and } \mathfrak{I}, i-1 \models \mathfrak{a}(\phi_1)\\
\textit{iff}\quad &\mathfrak{I}_\mathfrak{a}, i \models \mathsf{first} \text{ or } i > 0 \text{ and } \mathfrak{I}_\mathfrak{a}, i-1 \models f_{\phi_1}\\
\textit{iff}\quad &\mathfrak{I}_\mathfrak{a}, i \models \mathsf{first} \vee \mathsf{false}\,\mathsf{S}^< f_{\phi_1}.
\end{aligned}$$

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

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

iff there is some k, $i \leq k \leq n$, such that $\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2)$ and $\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1)$ for all j, $i \leq j < k$

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

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

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

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

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

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

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

All the other cases can be shown analogously. □

After this, f_ϕ is transformed into an equivalent LTL-formula f'_ϕ using the separation theorem from [52].

Proposition Appendix B.2 ([52]). *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 $\mathfrak{I}, i \models f'_\phi$.*

The formula f'_ϕ is in turn transformed into the LTL-formula f''_ϕ by replacing all variables that are in the scope an $\mathsf{U}^<$ by false and simplifying the resulting formula 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 the following lemma.

Lemma Appendix B.3. *For all $\mathfrak{I} = (\mathcal{I}_i)_{0\leq i\leq n}$ and $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \mathsf{N_C}$, we have $\mathfrak{I}_\mathfrak{a}, n \models f'_\phi$ iff $\mathfrak{I}_\mathfrak{a}, n \models f''_\phi$.*

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. $\mathfrak{I}_\mathfrak{a}$ only at time points $n' > n$. Thus, all of these occurrences can be replaced by false without affecting the value of f'_ϕ under $\mathfrak{I}_\mathfrak{a}$ at time point n. Furthermore, the equivalences listed above are clearly valid at any time point, and thus also do not affect this value. □

Finally, we transform f''_ϕ back into a TQ $\phi_{f''_\phi}$. This transformation is defined recursively as follows:

f''_ϕ	$\phi_{f''_\phi}$
p_j for j, $1 \leq j \leq m$	ψ_j
p	true
$f_1 \wedge f_2$	$\phi_{f_1} \wedge \phi_{f_2}$
$f_1 \vee f_2$	$\phi_{f_1} \vee \phi_{f_2}$
$\neg f_1$	$\neg \phi_{f_1}$
$f_1 \mathsf{S}^< f_2$	$\bigcirc^-(\phi_{f_2} \vee \phi_{f_1} \mathsf{S}^* \phi_{f_2})$

As before, we can show that the variable assignment $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \mathsf{N_C}$ is an answer to the Past-TQ $\phi_{f''_\phi}$ w.r.t. $\mathfrak{I}$ at time point i, $0 \leq i \leq n$, if and only if f''_ϕ is satisfied by $\mathfrak{I}_\mathfrak{a}$ at i.[2]

Lemma Appendix B.4. *For all $\mathfrak{I} = (\mathcal{I}_i)_{0 \leq i \leq n}$, all $\mathfrak{a}\colon \mathsf{FVar}(\phi) \to \mathsf{N_C}$, and all i, $0 \leq i \leq n$, we have $\mathfrak{I}_\mathfrak{a}, i \models f''_\phi$ iff $\mathfrak{I}, i \models \mathfrak{a}(\phi_{f''_\phi})$.*

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 $\mathfrak{I}_\mathfrak{a}, i \models p_j$ iff $\mathfrak{I}, i \models \mathfrak{a}(\psi_j)$ as in the proof of Lemma Appendix B.1. For p, we have $\mathfrak{I}_\mathfrak{a}, i \models p$ 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 $\wedge$, $\vee$, and $\neg$, the claim follows similarly as in the proof of Lemma Appendix B.1. It thus remains to show the claim for subformulae of the form $f_1 \mathsf{S}^< f_2$. We have

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

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

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

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

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

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

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

This finishes the reduction. Theorem 5.3 is now a simple consequence of the previous lemmata and the separation theorem.

Appendix C. Proofs for Section 6

Lemma 6.3. *The function Φ_0 is correct for 0.*

Proof. We prove by induction on the structure of the subqueries $\psi \in \mathsf{Sub}(\phi)$ 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 an atemporal query, then $\mathsf{eval}^n(\Phi_0(\psi)) = \mathsf{Ans}(\psi, \mathcal{I}_0) = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$.

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 $\mathsf{eval}^n(\Phi_0(\psi)) = \emptyset = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$; and $\mathsf{eval}^n(\Phi_0(\bullet^- \psi_1)) = \Delta^{\mathsf{N_V}} = \mathsf{Ans}(\bullet^- \psi_1, \mathfrak{I}^{(n)}, 0)$.

If $\psi = \psi_1 \mathsf{S} \psi_2$, then Proposition 3.4 yields that

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

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 for $n > 0$ we have, by Proposition 3.4,

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

and for $n = 0$ we get

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

Lemma 6.4. *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 the subqueries $\psi \in \mathsf{Sub}(\phi)$ 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

[2]Note that $\mathsf{FVar}(\phi_{f''_\phi}) = \mathsf{FVar}(\phi)$.

cases can easily be obtained from that of the corresponding cases in Lemma 6.3 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

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_{i-1}(\psi_1)) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i-1) \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

since Φ_{i-1} is correct for $i-1 < i \leq n$.

If $\psi = \psi_1 \,\mathsf{S}\, \psi_2$, then Proposition 3.4 yields that

$$\begin{aligned}&\mathsf{eval}^n(\Phi_i^0(\psi)) \\ &= \mathsf{eval}^n(\Phi_i^0(\psi_2)) \cup \big(\mathsf{eval}^n(\Phi_i^0(\psi_1)) \cap \mathsf{eval}^n(\Phi_{i-1}(\psi))\big) \\ &= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, i) \cup \big(\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cap \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i-1)\big) \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i). \qquad \square\end{aligned}$$

Lemma 6.5. *If Φ_{i-1} is correct for $i-1$ and $(i-1)$-bounded, then we can construct a function $\Phi_i \colon \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i$ that is correct for i and i-bounded.*

Proof. We substitute all variables from Var_{i-1}^ϕ by already computed answer terms of the form $\Phi_i^0(\psi)$. However, since these may themselves contain other variables from Var_{i-1}^ϕ, 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.

For this, we consider a total order $\psi^1 \prec \dots \prec \psi^k$ on the set $\mathsf{FSub}(\phi) = \{\psi^1, \dots, \psi^k\}$ with the property that whenever $\psi^j \in \mathsf{Sub}(\psi^{j'})$ for $j, j' \in \{1, \dots, k\}$, we have $j \leq j'$, i.e., $\psi^j = \psi^{j'}$ or $\psi^j \prec \psi^{j'}$. It is clear that such a total order exists and we fix one for the following considerations.

For $1 \leq j \leq k$ and $\psi \in \mathsf{Sub}(\phi)$, we obtain the answer term $\Phi_i^j(\psi)$ by replacing every occurrence of $x_{i-1}^{\psi^j}$ in $\Phi_i^{j-1}(\psi)$ with

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

Finally, we set $\Phi_i := \Phi_i^k$. It is easy to verify that each Φ_i^j is indeed a mapping from $\mathsf{Sub}(\phi)$ to AT_ϕ^i. Figure C.1 summarizes the process by which we obtain the families of answer terms Φ_i^j.

We now prove by induction on j that each Φ_i^j is correct for i. For $j = 0$, this is shown in Lemma 6.4. Consider now $j > 0$. Since eval^n is defined inductively on the structure of answer terms, it suffices to show that for all $n \geq i$, we have $\mathsf{eval}^n(x_{i-1}^{\psi^j}) = \mathsf{eval}^n(\mathsf{update}(x_{i-1}^{\psi^j}))$. For this, we make a case distinction on the form of ψ^j.

If $\psi^j = \bigcirc\psi_1$ or $\psi^j = \bullet\psi_1$, by definition 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 $\mathsf{eval}^n(\Phi_i^{j-1}(\psi_1)) = \mathsf{eval}^n(\mathsf{update}(x_{i-1}^{\psi^j}))$.

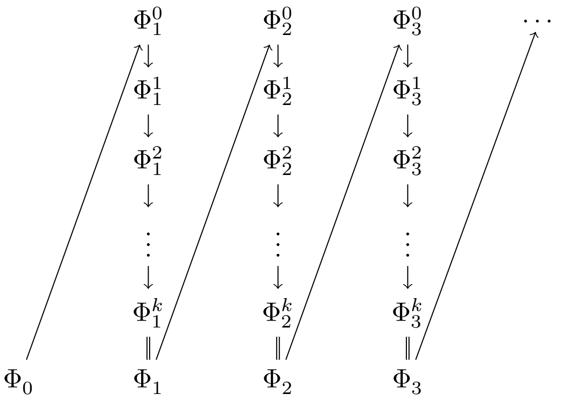

Figure C.1: The order in which the mappings Φ_i^j are computed

If $\psi^j = \psi_1 \,\mathsf{U}\, \psi_2$, 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 set as $\mathsf{eval}^n(\Phi_i^{j-1}(\psi^j)) = \mathsf{eval}^n(\mathsf{update}(x_{i-1}^{\psi^j}))$.

It remains to show i-boundedness of $\Phi_i = \Phi_i^k$, which we do by means of the following claim.

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

We prove this again by induction on j. For $j = 0$, we know from the definition of Φ_i^0 that for every $\psi \in \mathsf{Sub}(\phi)$ the answer term $\Phi_i^0(\psi)$ 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 $\mathsf{Var}_{i-1}^\psi \subseteq \{x_{i-1}^{\psi^1}, \dots, x_{i-1}^{\psi^k}\}$, and thus the claim is satisfied.

Let now $0 < j \leq k$ and assume that the claim holds for $j-1$. The function 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.

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

- If $\psi^j = \psi_1 \,\mathsf{U}\, \psi_2$, then $\mathsf{update}(x_{i-1}^{\psi^j}) = \Phi_i^{j-1}(\psi^j)$. Since Φ_i^{j-1} differs from Φ_i^0 only in the replacement of some of the variables with index $i-1$, 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 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 $\mathsf{Var}_{i-1}^{\psi_m} \cap \{x_{i-1}^{\psi^j}, \dots, x_{i-1}^{\psi^k}\}$. By similar arguments as above, the second set is actually empty.

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

Lemma 6.10. *There is a function $f\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ that is exponential in the first component and polynomial in the second such that we can compute each set $\mathsf{Ans}(\phi, \mathfrak{I}^{(n)})$, $n \geq 0$, in time at most $f(|\phi|, |\Delta|) + |\phi| \cdot t(|\phi|, |\Delta|)$ and space at most $f(|\phi|, |\Delta|) + s(|\phi|, |\Delta|)$.*

Proof. At each time point, we have to compute the sets $\mathsf{Ans}(\psi, \mathcal{I}_n)$ for all atemporal queries ψ occurring in ϕ, each time using $t(|\phi|, |\Delta|)$ time and $s(|\phi|, |\Delta|)$ space (which can be reused). These exponentially large sets then become the atoms of the new answer terms in Φ_n^0. These terms additionally contain answer terms $\Phi_{n-1}(\psi)$ (in normal form) for the previous time point, which are already exponential in $|\mathsf{N_V}|$ and $|\mathsf{FSub}(\phi)|$.

We have to be careful that the substitution process described in the proof of Lemma 6.5 does not introduce an additional exponential blowup in the size of ϕ. Each replacement step from Φ_n^{j-1} to Φ_n^j may replace exponentially many occurrences of the same variable by exponentially large n-bounded answer terms. However, by normalizing subterms $\Phi_n^j(\psi)$ that are already n-bounded after every such step, we can ensure that subsequent replacement steps again have to deal only with exponentially large replacement terms. This *local normalization* thus has to be done only for terms of the form given by the definition of Φ_n^0, where each component $\Phi_n^j(\psi)$ is already in normal form, and each component $\Phi_{n-1}(\psi)$ may contain exponentially many answer terms in normal form. This can be done in exponential time in $2|\mathsf{FSub}(\phi)|$ and $|\mathsf{N_V}|$.

Thus, we can compute (a normal form of) Φ_n with exponentially bounded resources. To compute $\mathsf{Ans}(\phi, \mathfrak{I}^{(n)})$, by Lemma 6.9 it now suffices to replace each variable by either $\emptyset$ or $\Delta^{\mathsf{N_V}}$ and evaluate the remaining set intersections and unions. □

Appendix D. Proofs for Section 7

Lemma 7.4. *Let $\mathcal{K} = \langle (\mathcal{A}_i)_{i\geq 0}, \mathcal{T}\rangle$ be a consistent infinite TKB. Then there is a set $\mathcal{R} \in \mathfrak{R}$ such that $\mathcal{K}_{\mathcal{R}}^{(n)}$ is consistent for all $n \geq 0$, and for every TCQ ϕ and all i and n with $0 \leq i \leq n$, we have*

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

Proof. 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 $P(c)$ with $P \in \mathsf{N_{RP}}$ that are entailed by the knowledge base $\langle \mathcal{A}_i \cup \mathcal{R}_j, \mathcal{T}\rangle$ for some $i \geq 0$, which results in a new set $\mathcal{R}_{j+1}$. We iterate this process until no new assertions are added. Let now $\mathcal{R}$ be the final set computed by this procedure 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. $\mathcal{K}_{\mathcal{R}}^{(n)}$ 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 and the fact that $\mathcal{K}_{\mathcal{R}}^{(n)}$ is consistent.

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 an index i, $0 \leq i \leq n$, with $\mathcal{I}_i \not\models \mathcal{R}$. Thus, there is $j > 0$ and $P(c) \in \mathcal{R}_j$ such that $\mathcal{I}_i \not\models P(c)$. Since $\mathfrak{I}$ respects the rigid predicates, this actually holds for all $i \geq 0$. By construction of $\mathcal{R}$, there must be an $i \geq 0$ such that $\langle \mathcal{A}_i \cup \mathcal{R}_{j-1}, \mathcal{T}\rangle$ entails $P(c)$. Since $\mathcal{I}_i \models \mathcal{A}_i$, $\mathcal{I} \models \mathcal{T}$, and $\mathcal{I}_i \not\models P(c)$, there must be an assertion $Q(d) \in \mathcal{R}_{j-1}$ such that $\mathcal{I}_i \not\models Q(d)$, which again holds for all $i \geq 0$. We can iterate this argument until we arrive at the fact that there must be an assertion $R(e) \in \mathcal{R}_0$ such that $\mathcal{I}_i \not\models R(e)$. This contradicts the fact that $\mathcal{R}_0 = \emptyset$. □

Theorem 7.5. *Let $\mathcal{Q}_1$, $\mathcal{Q}_2$ be query languages such that $\mathcal{Q}_1$ contains only rooted CQs, and $\mathcal{L}$ be a logic that has the canonical model property w.r.t. $\mathcal{Q}_1$ such that $\mathcal{Q}_1$-queries are $\mathcal{Q}_2$-rewritable w.r.t. $\mathcal{L}$. Let further $\mathcal{K} = \langle (\mathcal{A}_i)_{i\geq 0}, \mathcal{T}\rangle$ be a consistent infinite TKB and $\mathcal{R}$ given by Lemma 7.4.*

Then for all $n \geq 0$ there is a sequence of interpretations $\mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}} = (\mathcal{J}_i)_{0\leq i\leq n}$ such that for every temporal $\mathcal{Q}_1$-query ϕ, and all i, $0 \leq i \leq n$, we have

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

Proof. We start the construction of the sequence $\mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}$ with the canonical models $\mathcal{I}_i := \mathcal{I}_{\langle \mathcal{A}_i \cup \mathcal{R}, \mathcal{T}\rangle}$, $1 \leq i \leq n$, employed in Theorem 4.1 (but with $\mathcal{A}_i \cup \mathcal{R}$ instead of $\mathcal{A}_i$). By Definition 2.8, the domains $\Delta^{\mathcal{I}_i}$ of these canonical models are all countably infinite. We define the set $\mathcal{D} \subseteq 2^{\mathsf{N_{RP}}}$ of subsets of $\mathsf{N_{RP}}$ that contains exactly the sets

$$\rho(\mathcal{I}_i, x) := \{P \in \mathsf{N_{RP}} \mid x \in P^{\mathcal{I}_i}\}$$

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

To this end, consider i, n, $0 \leq i \leq n$, and $Y \in \mathcal{D}$. If $\mathcal{I}_i$ 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 $x \in \Delta^{\mathcal{I}_j}$ such that $Y = \rho(\mathcal{I}_j, x)$. To be on the safe side, we therefore construct the disjoint union $\mathcal{I}_i'$ of *all* interpretations in $\mathfrak{I}_{\mathcal{K}_{\mathcal{R}}^{(n)}}$ (with core $\mathcal{I}_i$).

To ensure that there are even countably infinitely many such individuals, we now define $\mathcal{I}_i''$ as the countably infinite disjoint union of $\mathcal{I}_i'$ with copies of itself (as core we take any of the copies). Finally, we ensure that all models have the same domain $\Delta := \mathsf{N_I} \cup (\mathcal{D} \times \mathbb{N})$ and interpret the constants 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 $\mathcal{J}_i$ and define $\mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}} := (\mathcal{J}_i)_{0 \le i \le n}$.

We now show that $\mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}$ is still a model of $\mathcal{K}_{\mathcal{R}}^{(n)}$. By our closure assumption on models in $\mathcal{L}$, the interpretations $\mathcal{J}_i$ 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 predicates on the constants was never changed. Furthermore, the sequence $\mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}$ respects the rigid predicates since the elements of $\mathcal{D} \times \mathbb{N}$ always satisfy exactly the rigid predicates given by their first component, and every $c \in \mathsf{N_C}$ satisfies at least the rigid predicates P for which $P(c) \in \mathcal{R}$. Assume now that we have $c^{\mathcal{J}_i} \in P^{\mathcal{J}_i}$ for some $P \in \mathsf{N_{RP}}$ and $c \in \mathsf{N_C}$, but $P(c) \notin \mathcal{R}$. By construction of $\mathcal{J}_i$, we thus also have $\mathcal{I}_i \models P(c)$. Since $\mathcal{I}_i$ is a canonical model of $\langle \mathcal{A}_i \cup \mathcal{R}, \mathcal{T} \rangle$ w.r.t. unary instance queries, all models of $\langle \mathcal{A}_i \cup \mathcal{R}, \mathcal{T} \rangle$ are also models of $P(c)$. But then we must have $P(c) \in \mathcal{R}$ by construction of $\mathcal{R}$ (see the proof of Lemma 7.4), which contradicts the assumption that $P(c) \notin \mathcal{R}$. This shows that every $c \in \mathsf{N_C}$ satisfies exactly the rigid predicates P with $P(c) \in \mathcal{R}$ in each $\mathcal{J}_i$.

Thus, $\mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}$ is a model of $\mathcal{K}_{\mathcal{R}}^{(n)}$ that respects the rigid predicates on the whole domain $\mathsf{N_I} \cup (\mathcal{D} \times \mathbb{N})$. This is the crucial property that allows us to show the first inclusion $\mathsf{Cert}(\phi, \mathcal{K}_{\mathcal{R}}^{(n)}, i) \subseteq \mathsf{Ans}(\phi, \mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}, i)$ exactly as in the proof of Theorem 4.1. Moreover, it directly follows from Theorem 4.1 that $\mathsf{Ans}(\phi^{\mathcal{T}}, \mathfrak{D}_{\mathcal{K}_{\mathcal{R}}^{(n)}}, i) \subseteq \mathsf{Cert}(\phi, \mathcal{K}_{\mathcal{R}}^{(n)}, i)$.

For the remaining inclusion, we again employ induction on the structure of ϕ. The only difference to the corresponding induction proof for Theorem 4.1 is the base case of a $\mathcal{Q}_1$-query; all the other cases can be shown as before. But for every $\mathcal{Q}_1$-query ψ, $\mathcal{Q}_2$-rewritability of $\mathcal{Q}_1$-queries w.r.t. $\mathcal{L}$ implies that

$$\begin{aligned}\mathsf{Ans}(\psi, \mathfrak{I}_{\mathcal{K}_{\mathcal{R}}^{(n)}}, i) &= \mathsf{Ans}(\psi, \mathcal{I}_i)\\ &= \mathsf{Ans}(\psi^{\mathcal{T}}, \mathcal{D}_{\langle \mathcal{A}_i \cup \mathcal{R}, \mathcal{T} \rangle})\\ &= \mathsf{Ans}(\psi^{\mathcal{T}}, \mathfrak{D}_{\mathcal{K}_{\mathcal{R}}^{(n)}}, i),\end{aligned}$$

and thus it suffices to show that $\mathsf{Ans}(\psi, \mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}, i)$ is a subset of $\mathsf{Ans}(\psi, \mathfrak{I}_{\mathcal{K}_{\mathcal{R}}^{(n)}}, i)$.

For this, consider any $\mathfrak{a} \in \mathsf{Ans}(\psi, \mathfrak{I}_{\mathcal{K}^{(n)},\mathcal{R}}, i)$. Thus, there exists a homomorphism π of $\mathfrak{a}(\psi)$ into $\mathcal{J}_i$, which can be used to define a homomorphism π' of $\mathfrak{a}(\psi)$ into $\mathcal{I}_i''$ by composition with the bijection between $\Delta^{\mathcal{I}_i''}$ and Δ (cf. Condition (ii) of Definition 2.5). Similarly, we obtain a homomorphism π'' of $\mathfrak{a}(\psi)$ into $\mathcal{I}_i'$ by taking for each $z \in \mathsf{Var}(\phi) \cup \mathsf{N_C}$ 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 $\mathfrak{a}(\psi)$ is rooted and the components in a disjoint union are not connected via the interpretation of predicates, the image of π'' must be contained in the original domain of $\mathcal{I}_i$. Thus, π'' is also a homomorphism of $\mathfrak{a}(\psi)$ into $\mathcal{I}_i$, i.e., we have $\mathfrak{a} \in \mathsf{Ans}(\psi, \mathfrak{I}_{\mathcal{K}_{\mathcal{R}}^{(n)}}, i)$. □