Stream Reasoning in Temporal Datalog

Alessandro Ronca, Mark Kaminski, Bernardo Cuenca Grau, Boris Motik and **Ian Horrocks**

Department of Computer Science, University of Oxford, UK

{alessandro.ronca, mark.kaminski, bernardo.cuenca.grau, boris.motik, ian.horrocks}@cs.ox.ac.uk

Abstract

In recent years, there has been an increasing interest in extending traditional stream processing engines with logical, rule-based, reasoning capabilities. This poses significant theoretical and practical challenges since rules can derive new information and propagate it both towards past and future time points; as a result, streamed query answers can depend on data that has not yet been received, as well as on data that arrived far in the past. Stream reasoning algorithms, however, must be able to stream out query answers as soon as possible, and can only keep a limited number of previous input facts in memory. In this paper, we propose novel reasoning problems to deal with these challenges, and study their computational properties on Datalog extended with a temporal sort and the successor function—a core rule-based language for stream reasoning applications.

1 Introduction

Query processing over data streams is a key aspect of Big Data applications. For instance, algorithmic trading relies on real-time analysis of stock tickers and financial news items (Nuti et al. 2011); oil and gas companies continuously monitor and analyse data coming from their wellsites in order to detect equipment malfunction and predict maintenance needs (Cosad et al. 2009); network providers perform real-time analysis of network flow data to identify traffic anomalies and DoS attacks (Münz and Carle 2007).

In stream processing, an input data stream is seen as an unbounded, append-only, relation of timestamped tuples, where timestamps are either added by the external device that issued the tuple or by the stream management system receiving it (Babu and Widom 2001; Babcock et al. 2002). The analysis of the input stream is performed using a standing query, the answers to which are also issued as a stream. Most applications of stream processing require near real-time analysis using limited resources, which poses significant challenges to stream management systems. On the one hand, systems must be able to compute query answers over the partial data received so far as if the entire (infinite) stream had been available; furthermore, they must stream query answers out with the minimum possible delay. On the other hand, due to memory limitations, systems can

only keep a limited *history* of previously received input facts in memory to perform computations. These challenges have been addressed by extending traditional database query languages with window constructs, which declaratively specify the finite part of the input stream relevant to the answers at the current time (Arasu, Babu, and Widom 2006).

In recent years, there has been an increasing interest in extending traditional stream management systems with logical, rule-based, reasoning capabilities (Barbieri et al. 2010; Calbimonte, Corcho, and Gray 2010; Anicic et al. 2011; Le-Phuoc et al. 2011; Zaniolo 2012; Özçep, Möller, and Neuenstadt 2014; Beck et al. 2015; Dao-Tran, Beck, and Eiter 2015). Rules can be very useful in stream processing applications for capturing complex analysis tasks in a declarative way, as well as for representing background knowledge about the application domain.

Example 1. *Consider a number of wind turbines scattered throughout the North Sea. Each turbine is equipped with a sensor, which continuously records temperature levels of key devices within the turbine and sends those readings to a data centre monitoring the functioning of the turbines. Temperature levels are streamed by sensors using a ternary predicate Temp, whose arguments identify the device, the temperature level, and the time of the reading. A monitoring task in the data centre is to track the activation of cooling measures in each turbine, record temperature-induced malfunctions and shutdowns, and identify parts at risk of future malfunction. This task is captured by the following set of rules:*

$$Temp(x, high, t) \rightarrow Flag(x, t) \tag{1}$$
$$Flag(x, t) \wedge Flag(x, t + 1) \rightarrow Cool(x, t + 1) \tag{2}$$
$$Cool(x, t) \wedge Flag(x, t + 1) \rightarrow Shdn(x, t + 1) \tag{3}$$
$$Shdn(x, t) \rightarrow Malfunc(x, t - 2) \tag{4}$$
$$Shdn(x, t) \wedge Near(x, y) \rightarrow AtRisk(y, t) \tag{5}$$
$$AtRisk(x, t) \rightarrow AtRisk(x, t + 1) \tag{6}$$

Rule (1) *'flags' a device whenever a high temperature reading is received. Rule* (2) *says that two consecutive flags on a device trigger cooling measures. Rule* (3) *says that an additional consecutive flag after activating cooling measures triggers a pre-emptive shutdown. By Rule* (4)*, a shutdown is due to a malfunction that occurred when the first flag leading to shutdown was detected. Finally, Rules* (5) *and* (6) *identify*

devices located near a shutdown device as being at risk and propagate risk recursively into the future.

The power and flexibility provided by rules poses additional challenges. As seen in our example, rules can derive information and propagate it both towards past and future time points. As a result, query answers can depend on data that has not yet been received (thus preventing the system from streaming out answers as soon as new input arrives), as well as on data that arrived far in the past (thus forcing the system to keep in memory a potentially large input history).

Towards developing a solid foundation for rule-based stream reasoning, we propose in Section 3 a suite of decision problems that can be exploited by a stream reasoning algorithm to deal with the aforementioned challenges.

- The *definitive time point* (DTP) problem is to check whether query answers to be issued at a given time τ_{out} will remain unaffected by any future input data given the current history; if so, τ_{out} is *definitive* and answers at τ_{out} can be safely output by the algorithm.

- The *forgetting* problem is to determine whether facts received at a given previous time point and recorded in the current history can be 'forgotten', in that they cannot affect future query answers. Forgetting allows the algorithm to maintain as small a history as possible.

- The *delay* problem is to check, given a time gap d, whether time point $\tau_{in} - d$ is definitive for each time point τ_{in} at which new input facts are received and each history up to τ_{in}. Delay can thus be seen as a data-independent variant of DTP: the delay d can be computed offline before receiving any data, and the algorithm can then safely output answers at $\tau_{in} - d$ as data at τ_{in} is being received.

- The *window size* problem is a data-independent variant of forgetting. The task is to determine, given a window size s, whether all history facts at time points up to $\tau_{in} - s$ can be forgotten for each time τ_{in} at which new input facts are received and each history up to τ_{in}. A stream reasoning algorithm can compute s in an offline phase and then, in the online phase, immediately delete all history facts older than s time points as new data arrives.

In Section 4, we proceed to the study of the computational properties of the aforementioned problems. For this, we consider as query language *temporal Datalog*—negation-free Datalog with a special temporal sort to which the successor function (or, equivalently, addition by a constant) is applicable (Chomicki and Imieliński 1988). This is a core temporal rule-based language, which captures other prominent temporal languages (Abadi and Manna 1989; Baudinet, Chomicki, and Wolper 1993) and forms the basis of more expressive formalisms for stream reasoning recently proposed in the literature (Zaniolo 2012; Beck et al. 2015).

We show in Section 4.1 that DTP is PSPACE-complete in data complexity and becomes tractable for nonrecursive queries under very mild additional restrictions; thus, DTP is no harder than query evaluation (Chomicki and Imieliński 1988). In Section 4.2, we show that forgetting is undecidable; however, quite surprisingly, the aforementioned restrictions to nonrecursive queries allows us to regain not

only decidability, but also tractability in data complexity. In Section 4.3, we turn our attention to data-independent problems. We show that both delay and window size are undecidable in general and become co-NEXP-complete for nonrecursive queries.

Our results show that, although stream reasoning problems are either intractable in data complexity or undecidable in general, they become feasible in practice for nonrecursive queries under very mild additional restrictions. On the one hand, the data-dependent problems (DTP and forgetting) become tractable in data complexity (a very important requirement for achieving near real-time computation in practice); on the other hand, although the data-independent problems (delay and window size) remain intractable, these are one-time problems which only need to be solved once prior to receiving any input data.

The proofs of all results are given in the appendix of this paper.

2 Preliminaries

Syntax A vocabulary consists of *predicates*, *constants* and *variables*, where constants are partitioned into *objects* and integer *time points* and variables are partitioned into *object variables* and *time variables*. An *object term* is an object or an object variable. A *time term* is either a time point, a time variable, or an expression of the form $t + k$ where t is a time variable, k is an integer number, and $+$ is the standard *integer addition function*. The offset $\Delta(s)$ of a time term s equals zero if s is a time variable or a time point and it equals k if s is of the form $s = t + k$.

Predicates are partitioned into *extensional* (EDB) and *intensional* (IDB) and they come with a nonnegative integer *arity* n, where each position $1 \leq i \leq n$ is of either *object* or *time sort*. A predicate is *rigid* if all its positions are of object sort and it is *temporal* if the last position is of time sort and all other positions are of object sort. An *atom* is an expression $P(t_1, \ldots, t_n)$ where P is a predicate and each t_i is a term of the required sort. A *rigid* atom (respectively, temporal, IDB, EDB) is an atom involving a rigid predicate (respectively, temporal, IDB, EDB).

A *rule* r is of the form $\bigwedge_i \alpha_i \rightarrow \alpha$, where α and each α_i are rigid or temporal atoms, and α is IDB whenever $\bigwedge_i \alpha_i$ is non-empty. Atom $\mathrm{head}(r) = \alpha$ is the *head* of r, and $\mathrm{body}(r) = \bigwedge_i \alpha_i$ is the *body* of r. Rules are assumed to be *safe*: each head variable must occur in the body. An *instance* r' of r is obtained by applying a substitution to r. A *program* Π is a finite set of rules. Predicate P is Π-*dependent* on a predicate P' if there is a rule of Π with P in the head and P' in the body. The rank $\mathrm{rank}(P, \Pi)$ of P w.r.t. Π is 0 if P does not occur in head position in Π, and is the maximum of the values $\mathrm{rank}(P') + 1$ for P' a predicate such that P is Π-dependent on P' otherwise. We write $\mathrm{rank}(P)$ for $\mathrm{rank}(P, \Pi)$ if Π is clear from the context. The rank $\mathrm{rank}(\Pi)$ of Π is the maximum rank of a predicate in Π.

A *query* is a pair $Q = \langle P_Q, \Pi_Q \rangle$ where Π_Q is a program and P_Q is an IDB predicate in Π_Q; query Q is *temporal* (*rigid*) if P_Q is a temporal (rigid) predicate. A term, atom, rule, or program is *ground* if it contains no variables. A *fact* α is a ground, function-free rigid or temporal atom; every

fact α corresponds to a rule of the form $\top \to \alpha$ where $\top$ denotes the empty conjunction, so we use α and $\top \to \alpha$ interchangeably. A *dataset* D is a program consisting of EDB facts. The τ-*segment* $D[\tau]$ of dataset D is the subset of D containing all rigid facts and all temporal facts with time argument $\tau' > \tau$.

A program Π (respectively, query Q) is: *Datalog* if no temporal predicate occurs in Π (in Π_Q); and *nonrecursive* if the directed graph induced by the Π-dependencies (Π_Q-dependencies) is acyclic.

Semantics and standard reasoning Rules are interpreted in the standard way as universally quantified first-order sentences. A Herbrand interpretation $\mathcal{H}$ is a (possibly infinite) set of facts. Interpretation $\mathcal{H}$ *satisfies* a rigid atom α if $\alpha \in \mathcal{H}$, and it satisfies a temporal atom α if evaluating the addition function in α yields a fact in $\mathcal{H}$. The notion of satisfaction is extended to conjunctions of ground atoms, rules and programs in the standard way. If $\mathcal{H} \models \Pi$, then $\mathcal{H}$ is a *model* of Π. Program Π *entails* a fact α, written $\Pi \models \alpha$, if $\mathcal{H} \models \Pi$ implies $\mathcal{H} \models \alpha$. The *answers* to a query Q over a dataset D, written $Q(D)$, are the tuples $\mathbf{a}$ of constants such that $\Pi_Q \cup D \models P_Q(\mathbf{a})$. If Q is a temporal query, we denote with $Q(D, \tau)$ the subset of answers in $Q(D)$ referring to time point τ. Given an input query Q, dataset D and tuple $\mathbf{a}$, the *query evaluation problem* is to check whether $\mathbf{a}$ is an answer to Q over D; the *data complexity* of query evaluation is the complexity when Q is considered fixed. Finally, a query Q_1 is *contained* in a query Q_2, written $Q_1 \sqsubseteq Q_2$, if $Q_1(D) \subseteq Q_2(D)$ for every dataset D. Given input queries Q_1 and Q_2, the *query containment problem* is to check whether Q_1 is contained in Q_2.

Complexity Query evaluation is PSPACE-complete in data complexity assuming that numbers are coded in unary (Chomicki and Imieliński 1988). Data complexity drops to the circuit class AC^0 for nonrecursive programs. By standard results in nontemporal Datalog, containment of temporal queries is undecidable (Shmueli 1993). Furthermore, it is co-NExp-hard for nonrecursive queries (Benedikt and Gottlob 2010).

3 Stream Reasoning Problems

A stream reasoning algorithm receives as input a query Q and a stream of temporal EDB facts, and produces as output a stream of answers to Q. Both input facts and query answers are processed by increasing value of their timestamps, where τ_{in} and τ_{out} represent the current times at which input facts are received and query answers are being streamed out, respectively. Answers at τ_{out} are only output when the algorithm can determine that they cannot be affected by future input facts. In turn, input facts received so far are kept in a *history dataset* D since future query answers can be influenced by facts received at an earlier time; practical systems, however, have limited memory and hence the algorithm must also *forget* facts in the history as soon as it can determine that they will not influence future query answers.

Algorithms 1 and 2 provide two different realisations of such a stream reasoning algorithm, which we refer to as *online* and *offline*, respectively.

Algorithm 1: 'Online' Stream Reasoning Algorithm

Parameters: Temporal query Q

1   $D := \emptyset$, $\tau_{in} := 0$, $\tau_{out} := 0$, $\tau_{mem} := 0$
2   **loop**
3   receive facts U that hold at τ_{in} and set $D := D \cup U$
4   **while** $\tau_{out} \leq \tau_{in}$ **and** τ_{out} *is definitive for* Q, D, τ_{in}
      do
5   stream output $Q(D, \tau_{out})$
6         $\tau_{out} := \tau_{out} + 1$
7   **end**
8   **while** $\tau_{mem} < \tau_{out}$ **and**
      τ_{mem} *is forgettable for* $Q, D, \tau_{in}, \tau_{out}$ **do**
9   forget all temporal facts in D holding at τ_{mem}
10        $\tau_{mem} := \tau_{mem} + 1$
11   **end**
12      $\tau_{in} := \tau_{in} + 1$
13 **end**

The *online algorithm* (see Algorithm 1) decides which answers to stream and which history facts to forget 'on the fly' as new input data arrives. The algorithm records the latest time point τ_{out} for which answers have not yet been streamed; as τ_{in} increases and new data arrives, the algorithm checks (lines 4-7) whether answers at τ_{out} can now be streamed and, if so, it continues incrementing τ_{out} until it finds a time point for which answers cannot be provided yet. This process relies on deciding whether the considered τ_{out} are *definitive*—that is, the answers to Q at τ_{out} for the history D will remain stable even if D were extended with an unknown (and thus arbitrary) set U of future input facts.

Definition 1. *A τ_{in}-history D is a dataset consisting of rigid facts and temporal facts with time argument at most τ_{in}. A τ_{in}-update U is a dataset consisting of temporal facts with time argument strictly greater than τ_{in}.*

Definition 2. *An instance I of the* Definitive Time Point *(DTP) problem is a tuple $\langle Q, D, \tau_{in}, \tau_{out} \rangle$, with Q a temporal query, D a τ_{in}-history and $\tau_{out} \leq \tau_{in}$. DTP holds for I iff $Q(D, \tau_{out}) = Q(D \cup U, \tau_{out})$ for each τ_{in}-update U.*

Example 2. *Consider Example 1, and suppose we are interested in determining the time points at which a turbine malfunctions. Thus, let the query Q have output predicate $Malfunc$ and include rules (1)–(4) together with rule*

$$Temp(x, na, t) \to Malfunc(x, t)$$

which defines an invalid reading as a malfunction. For a history D consisting of the fact $Temp(a, high, 0)$, we have that $DTP(Q, D, 0, 0)$ is false, since $Q(D, 0)$ is empty and $Q(D \cup U, 0)$ is not if the update U contains $Temp(a, high, 1)$ and $Temp(a, high, 2)$. For a history D' consisting of the fact $Temp(a, na, 0)$, we have that $DTP(Q, D', 0, 0)$ is true, since $Q(D', 0)$ already includes the only possible answer.

Algorithm 1 also records the latest time point τ_{mem} for which history facts have not yet been forgotten. As τ_{in} increases, the algorithm checks in lines 8-11 whether all history facts at time τ_{mem} can now be forgotten and, if so, it continues incrementing τ_{mem} until it finds a point where this is no longer possible. For this, the algorithm decides whether

---
Algorithm 2: 'Offline' Stream Reasoning Algorithm

Parameters: Temporal query Q
1 $D := \emptyset$, $\tau_{in} := 0$
2 compute minimal delay d and minimal window size s for Q
3 **loop**
4   receive facts U that hold at τ_{in} and set $D := D \cup U$
5   **if** $\tau_{in} - d \geq 0$ **then** stream output $Q(D, \tau_{in} - d)$
6   forget all temporal facts in D holding at $\tau_{in} - s$
7      $\tau_{in} := \tau_{in} + 1$
8 **end**
---

the relevant τ_{mem} are *forgettable*, in the sense that no future answer to Q can be affected by the history facts at τ_{mem}.

Definition 3. *An instance I of* FORGET *is a tuple of the form* $\langle Q, D, \tau_{in}, \tau_{out}, \tau_{mem} \rangle$, *with Q a temporal query, D a τ_{in}-history, and $\tau_{mem} \leq \tau_{out} \leq \tau_{in}$.* FORGET *holds for I iff $Q(D \cup U, \tau) = Q(D[\tau_{mem}] \cup U, \tau)$ for each τ_{in}-update U and each time point $\tau \geq \tau_{out}$.*

Example 3. *Consider the query Q with output predicate $Shdn$ and rules* (1)–(3) *from Example 1. For a history D consisting of facts $Temp(a, high, 0)$ and $Temp(a, low, 1)$, we have that* FORGET$(Q, D, 1, 1, 1)$ *is true, since $Q(D[1] \cup U, 1)$ is empty for every 1-update U. For a history D' containing facts $Temp(a, high, 0)$ and $Temp(a, high, 1)$, we have that* FORGET$(Q, D, 1, 1, 1)$ *is false, since $Q(D[1] \cup U, 1)$ is empty but $Q(D \cup U, 1)$ is not for the 1-update containing $Temp(a, high, 2)$.*

The *offline algorithm* (Algorithm 2) precomputes the minimum *delay d* and *window size s* for the standing query Q in a way that is independent from the input data stream.

Intuitively, d represents the smallest time gap needed to ensure that, for any input stream and any time point τ_{in}, the time point $\tau_{out} = \tau_{in} - d$ is definitive; in other words, that it is always safe to stream answers with a delay d relative to the currently processed input facts.

Definition 4. *An instance I of* DELAY *is a pair $\langle Q, d \rangle$, with Q a temporal query and d a nonnegative integer.* DELAY *holds for I iff $Q(D, \tau_{in} - d) = Q(D \cup U, \tau_{in} - d)$ for each time point τ_{in}, each τ_{in}-history D, and each τ_{in}-update U.*

Example 4. *In Example 1, 0 is a valid delay for the AtRisk and Shdn queries, and so is 2 for the Malfunc query.*

In turn, s represents the size of the smallest time interval for which the history needs to be kept; in other words, for any input stream and any time point τ_{in}, it is safe to forget all history facts with timestamp smaller than $\tau_{in} - s$.

Definition 5. *An instance I of* WINDOW *is a triple $\langle Q, d, s \rangle$, with Q a temporal query and d and s nonnegative integers.* WINDOW *holds for I iff $Q(D \cup U, \tau_{out}) = Q(D[\tau_{in} - s] \cup U, \tau_{out})$ for all time points τ_{in} and τ_{out} with $\tau_{out} > \tau_{in} - d$, each τ_{in}-history D, and each τ_{in}-update U.*

Example 5. *Consider Example 1. Assuming that we want to evaluate queries with delay 0, a valid window size for the Shdn query is 2; whereas the AtRisk query has no valid window size (or, equivalently, the query requires a window of infinite size), since answers for that query can depend on facts arbitrarily far in the past.*

Once the delay d and window size s have been determined, they remain fixed during execution of the algorithm: indeed, as τ_{in} increases and new data arrives in each iteration of the main loop, Algorithm 2 simply streams query answers at $\tau_{in} - d$ and forgets all history facts at $\tau_{in} - s$. This is in contrast to the online approach, where the algorithm had to decide in each iteration of the main loop which answers to stream and which facts to forget.

4 Complexity of Stream Reasoning

We now start our investigation of the computational properties of the stream reasoning problems introduced in Section 3. For all problems, we consider both the general case applicable to arbitrary inputs and the restricted setting where the input queries are nonrecursive. All our results assume that numbers are coded in unary.

4.1 Definitive Time Point

Let $I = \langle Q, D, \tau_{in}, \tau_{out} \rangle$ be a fixed, but arbitrary, instance of DTP and denote with Ω_I the set consisting of all objects in $\Pi_Q \cup D$ and a fresh object o_I unique to I.

As stated in Definition 2, DTP holds for I if and only if the query answers at time τ_{out} over the history D coincide with the answers at the same time point but over D extended with an arbitrary τ_{in}-update U. Note that, in addition to new facts over existing objects in D and Q, the update U may also include facts about new objects. The following proposition shows that, to decide DTP, it suffices to consider updates involving only objects from Ω_I. Intuitively, updates containing fresh objects can be homomorphically embedded into updates over Ω_I by mapping all fresh objects to o_I.

Proposition 1. DTP *holds for I iff $\mathbf{o} \in Q(D \cup U, \tau_{out})$ implies $\mathbf{o} \in Q(D, \tau_{out})$ for every tuple $\mathbf{o}$ over Ω_I and every τ_{in}-update U involving only objects in Ω_I.*

The general case. We next show that DTP is decidable and provide tight complexity bounds. Our upper bounds are obtained by showing that, to decide DTP, it suffices to consider a single *critical update* and a slight modification of the query, which we refer to as the *critical query*. Intuitively, the critical update is a dataset that contains all possible facts at the next time point $\tau_{in} + 1$ involving EDB predicates from Q and objects in Ω_I. In turn, the critical query extends Q with rules that propagate all facts in the critical update recursively into the future. The intention is that the answers to the critical query over D extended with the critical update will capture the answers to Q over D extended with any arbitrary future update. In the following definition, we use ψ to denote the renaming mapping each temporal EDB predicate to a fresh temporal IDB predicate of the same arity.

Definition 6. *Let A be a fresh unary temporal EDB predicate. The* critical update Υ_I *for I is the τ_{in}-update containing the fact $A(\tau_{in} + 1)$, and all facts $P(\mathbf{o}, \tau_{in} + 1)$ for each temporal EDB predicate P in Π_Q and each tuple $\mathbf{o}$ over Ω_I.*

Let V be a fresh unary temporal IDB predicate. The critical query Θ_I *for I is the query where $P_{\Theta_I} = P_Q$ and Π_{Θ_I} is obtained from $\psi(\Pi_Q)$ by adding rule $A(t) \to V(t)$, rule*

$V(t) \rightarrow V(t+1)$, *and the following rules for each temporal EDB predicate P occurring in* Π_Q*, where* $P' = \psi(P)$:

$$P(\mathbf{x}, t) \rightarrow P'(\mathbf{x}, t)$$
$$V(t+1) \wedge P'(\mathbf{x}, t) \rightarrow P'(\mathbf{x}, t+1)$$

The construction of the critical query and update ensures, on the one hand, that $Q(D, \tau_{out}) = \Theta_I(D, \tau_{out})$ and, on the other hand, that $Q(D \cup U, \tau_{out}) \subseteq \Theta_I(D \cup \Upsilon_I, \tau_{out})$ for each τ_{in}-update U involving only objects in Ω_I. We can exploit these properties, together with Proposition 1, to show that DTP can be decided by checking whether, at τ_{out}, the answers to the critical query over D remain the same if D is extended with the critical update.

Lemma 1. DTP *holds for I iff* $\mathbf{o} \in \Theta_I(D \cup \Upsilon_I, \tau_{out})$ *implies* $\mathbf{o} \in \Theta_I(D, \tau_{out})$ *for every tuple* $\mathbf{o}$ *over* Ω_I.

It follows from Lemma 1 that, to decide DTP, we need to perform two temporal query evaluation tests for each candidate tuple $\mathbf{o}$. Since temporal query evaluation is feasible in PSPACE in data complexity, then so is DTP because the number of candidate tuples $\mathbf{o}$ is polynomial if Q is fixed.

Furthermore, query evaluation is reducible to DTP, and hence the aforementioned PSPACE upper bound in data complexity is tight.

Theorem 1. DTP *is* PSPACE-complete *in data complexity*.

Nonrecursive queries We next show that DTP becomes tractable in data complexity for nonrecursive queries. In the remainder of this section, we fix an arbitrary instance $I = \langle Q, D, \tau_{in}, \tau_{out} \rangle$ of DTP, where Q is nonrecursive. We assume w.l.o.g. that Q does not contain rigid atoms: each such atom $P(\mathbf{c})$ can be replaced with a temporal atom of the form, e.g., $P'(\mathbf{c}, 0)$. We make the additional technical assumption that each rule in Q is restricted as follows.

Definition 7. *A rule is* connected *if it contains at most one temporal variable, which occurs in the head whenever it occurs in the body. A query is* connected *if so are its rules.*

Restricting our arguments to connected queries allows us to considerably simplify definitions and proofs.

We start with the observation that the critical query of I always includes recursive rules that propagate information arbitrarily far into the future (see Definition 6). As a result, our general algorithm for DTP does not immediately provide an improved upper bound in the nonrecursive case.

The need for such recursive rules, however, is motivated by the fact that the answers to a recursive query Q at time τ_{out} may depend on facts at time points τ arbitrarily far from τ_{out}; in other words, there is no bound b for I such that $|\tau - \tau_{out}| \leq b$ in every derivation of query answers at τ_{out} involving input facts at τ. If Q is nonrecursive, however, such a bound is guaranteed to exist and can be established based on the following notion of program *radius*. Intuitively, query answers at $\tau_{out} \leq \tau_{in}$ can only be influenced by future facts whose timestamp is located within the interval $[\tau_{in}, \tau_{out} + \mathrm{rad}(\Pi_Q)]$.

Definition 8. *Let r be a connected rule mentioning a time variable. The* radius $\mathrm{rad}(r)$ *of r is the maximum of the values* $|\Delta(s) - \Delta(s')|$ *for s the time argument in the head of r and*

s' *the time argument in a body atom of r. The* radius $\mathrm{rad}(\Pi)$ *of a connected program Π is given by the number of rules in Π multiplied by the maximum radius of a rule in Π.*

Thus, to show tractability of DTP, we identify a polynomially bounded number of *critical time points* using the radius of Π_Q and argue that we can dispense with the aforementioned recursive rules by constructing a critical update for I that includes all facts over these time points. Since Π_Q may contain explicit time points in rules, these also need to be taken into account when defining the relevant critical time points and the corresponding critical update.

Definition 9. *Let τ_0 be the maximum value between τ_{out} and the largest time point occurring in* Π_Q.

A time point τ is critical I *if $\tau_{in} < \tau \leq \tau_0 + \mathrm{rad}(\Pi_Q)$. The* bounded critical update Υ_I^{b} *of I consists of each fact $P(\mathbf{o}, \tau)$ with P a temporal EDB predicate in Π_Q, $\mathbf{o}$ a tuple over Ω_I, and τ a critical time point.*

The following lemma justifies the key property of the critical update Υ_I^{b}, namely that the answers to Q over $D \cup \Upsilon_I^{\mathrm{b}}$ capture those over D extended with any future update.

Lemma 2. *Let a be the maximum radius of a rule in Π_Q and let $\mathbf{T}$ consist of τ_{out} and the time points in Π_Q. If $\Pi_Q \cup D \cup U \models P(\mathbf{o}, \tau)$ for a τ_{in}-update U involving only objects in Ω_I and a predicate P in Π_Q, and $|\tau - \tau'| \leq a \cdot (\mathrm{rank}(\Pi_Q) - \mathrm{rank}(P))$ for some $\tau' \in \mathbf{T}$, then $\Pi_Q \cup D \cup \Upsilon_I^{\mathrm{b}} \models P(\mathbf{o}, \tau)$.*

We are now ready to establish the analogue to Lemma 1 in the nonrecursive case.

Lemma 3. DTP *holds for I iff* $\mathbf{o} \in Q(D \cup \Upsilon_I^{\mathrm{b}}, \tau_{out})$ *implies* $\mathbf{o} \in Q(D, \tau_{out})$ *for every tuple* $\mathbf{o}$ *over* Ω_I.

Tractability of DTP then follows from Lemma 3 and the tractability of query evaluation for nonrecursive programs.

Theorem 2. DTP *is in* P *in data complexity if restricted to nonrecursive connected queries.*

4.2 Forgetting

We now move on to the forgetting problem as given in Definition 3. Unfortunately, in contrast to DTP, forgetting is undecidable. This follows by a reduction from containment of nontemporal Datalog queries—a well-known undecidable problem (Shmueli 1993).

Theorem 3. FORGET *is undecidable.*

In the remainder of this section we show that, by restricting ourselves to nonrecursive input queries, we can regain not only decidability of forgetting, but also tractability in data complexity. Let $I = \langle Q, D, \tau_{in}, \tau_{out}, \tau_{mem} \rangle$ be an arbitrary instance of FORGET where Q is nonrecursive. We adopt the same technical assumptions as in Section 4.1 for DTP in the nonrecursive case. Additionally, we assume that Q does not contain explicit time points in rules—note that the rules in our running example satisfy this restriction. We believe that dropping this assumption does not affect tractability, but we leave this question open for future work.

By Definition 3, to decide FORGET for I, we must check whether the answers $Q(D \cup U, \tau)$ are included in $Q(D[\tau_{mem}] \cup U, \tau)$ for every τ_{in}-update U and $\tau \geq \tau_{out}$.

Similarly to the case of DTP, we identify two time intervals of polynomial size in data, and show that it suffices to consider only updates U over the first interval and only time points τ over the second interval. In contrast to DTP, however, we need to potentially consider all possible such updates and cannot restrict ourselves to a single critical one.

Note, however, that checking the aforementioned inclusion of query answers for all relevant updates and time points would lead to an exponential blowup in data complexity. To overcome this, we define instead nonrecursive queries Q_1 and Q_2 such that the desired condition holds if and only if $Q_1 \sqsubseteq Q_2$, where Q_1 and Q_2 contain a (fixed) rule set derived from Q and a portion of the history D. Then, we show that checking such containment where only the data-dependent rules are considered part of the input is feasible in polynomial time.

We start by identifying the set of relevant time points for query answers and updates.

Definition 10. *A time point τ is* output-relevant *for I if $\tau_{out} \leq \tau \leq \tau_{mem} + \mathrm{rad}(Q)$. In turn, it is* update-relevant *for I if it satisfies $\tau_{in} < \tau \leq \tau_{mem} + 2 \cdot \mathrm{rad}(Q)$.*

Intuitively, answers at time points bigger than $\tau_{mem} + \mathrm{rad}(Q)$ cannot be affected by history facts that hold before τ_{mem}; thus, we do not need to consider answers at time points that are not output-relevant. In turn, output-relevant time points cannot depend on facts in an update after $\tau_{mem} + 2 \cdot \mathrm{rad}(Q)$; thus, it suffices to consider updates containing only facts that hold at update-relevant time points. We next construct the aforementioned queries Q_1 and Q_2 using the identified update-relevant and output-relevant time points.

Definition 11. *Let B be a fresh temporal IDB unary predicate and let D_0 consist of all facts $B(\tau)$ for each update-relevant time point τ. For ψ as in Definition 6, let Π be the smallest program containing:* (i) *each rule in $\psi(\Pi_Q)$ having a predicate different from P_Q in the head;* (ii) *each rule obtained by grounding the time argument to an output-relevant time point in a rule in $\psi(\Pi_Q)$ having P_Q as head predicate; and* (iii) *the rule $P(\mathbf{x},t) \wedge B(t) \rightarrow P'(\mathbf{x},t)$ for each temporal EDB predicate P occurring in Π_Q, where $P' = \psi(P)$.*

We now let $Q_1 = \langle P_Q, \Pi_1 \rangle$ and $Q_2 = \langle P_Q, \Pi_2 \rangle$, where $\Pi_1 = \Pi \cup D_0 \cup \psi(D)$, and $\Pi_2 = \Pi \cup D_0 \cup \psi(D[\tau_{mem}])$.

Intuitively, the facts about B are used to 'tag' the update-relevant time points; rule $P(\mathbf{x},t) \wedge B(t) \rightarrow P'(\mathbf{x},t)$, when applied to the history D and any update U, will 'project' U to the relevant time points and filter out all facts in D. Finally, the rules in *(i)* and *(ii)* allow us to derive the same consequences (modulo predicate renaming) as Π_Q, but only over the output-relevant time points.

We can now establish correctness of our approach.

Lemma 4. FORGET *holds for I iff $Q_1 \sqsubseteq Q_2$, where Q_1 and Q_2 are as given in Definition 11.*

Theorem 4. FORGET *is in* P *in data complexity if restricted to nonrecursive connected queries whose rules contain no time points.*

This concludes our discussion of the data-dependent problems motivated by our 'online' stream reasoning algorithm (recall Algorithm 1). In the following section, we turn our attention to the data-independent problems motivated by our 'offline' approach (Algorithm 2).

4.3 Data-Independent Problems

Query containment can be reduced to both DELAY and WINDOW using a variant of the reduction we used in Section 3 for the forgetting problem. As a result, we can show undecidability of both of our data-independent reasoning problems in the general case.

Theorem 5. DELAY *is undecidable.*

Theorem 6. WINDOW *is undecidable.*

Furthermore, our reductions from query containment preserve the shape of the queries and hence they also provide a co-NExp lower bound for both problems in the nonrecursive case (Benedikt and Gottlob 2010). In the remainder of this section, we show that this bound is tight.

The co-NExp upper bounds are obtained via reductions from DELAY and WINDOW into query containment for temporal Datalog, which we detail in the remainder of this section. Similarly to previous sections, we assume that queries are connected and also that they do not contain explicit time points or objects; the latter restriction is consistent with the 'purity' assumption in (Benedikt and Gottlob 2010).

Note, however, that the upper bound in (Benedikt and Gottlob 2010) for query containment only holds for standard nonrecursive Datalog; therefore, we first establish that this upper bound extends to the temporal case. Intuitively, temporal queries can be transformed into nontemporal ones by grounding them to a finite number of relevant time points based on their (finite) radius.

Lemma 5. *Query containment restricted to nonrecursive queries that are connected and constant-free is in* co-NExp.

We now proceed to discussing our reductions from DELAY and WINDOW into temporal query containment.

Consider a fixed, but arbitrary, instance $I = \langle Q, d \rangle$ of DELAY. We construct queries Q_1 and Q_2 providing the basis for our reduction.

Let A and B be fresh fresh unary temporal predicates, where A is EDB and B is IDB. Furthermore, let G be a fresh temporal IDB predicate of the same arity as P_Q. Let Π_1 extend Π_Q with the following rule:

$$P_Q(\mathbf{x},t) \wedge A(t) \rightarrow G(\mathbf{x},t) \tag{7}$$

and let $Q_1 = \langle G, \Pi_1 \rangle$. Intuitively, Q_1 restricts the answers to Q to time points where A holds.

Let $Q_2 = \langle G, \Pi_2 \rangle$, where Π_2 is now the program obtained from $\psi(\Pi_Q)$ by adding the previous rule (7) and the following rules for each k satisfying $-\mathrm{rad}(Q) \leq k \leq d$ and each temporal EDB predicate P in Π_Q, where $P' = \psi(P)$:

$$A(t) \rightarrow B(t+k) \tag{8}$$
$$P(\mathbf{x},t) \wedge B(t) \rightarrow P'(\mathbf{x},t) \tag{9}$$

Intuitively, Q_2 further restricts the answers to Q_1 at any time τ_{out} to those that can be derived using facts in the interval $[\tau_{out} - \mathrm{rad}(Q), \tau_{out} + d]$.

It then follows that $Q_1 \sqsubseteq Q_2$ if and only if DELAY holds for I. Furthermore, the construction of Q_1 and Q_2 is feasible in LOGSPACE.

Theorem 7. DELAY *restricted to nonrecursive queries that are connected and constant-free is* co-NEXP-*complete.*

We conclude by providing the upper bound for WINDOW. For this, consider an arbitrary instance $I = \langle Q, d, s \rangle$. Similarly to the case of DELAY, we construct queries Q_1 and Q_2 for which containment holds iff WINDOW holds for I. Let In, Out and B be fresh unary temporal predicates, where In is EDB and Out, B are IDB. Furthermore, as before, let G be a fresh temporal IDB predicate of the same arity as P_Q.

For j a nonnegative integer, let Π^j be the program extending $\psi(\Pi_Q)$ with the following rules for each k satisfying $-d < k \leq -s + \mathrm{rad}(Q)$, each ℓ satisfying $-j < \ell \leq -s + 2 \cdot \mathrm{rad}(Q)$, and each temporal EDB predicate P in Π_Q, where $P' = \psi(P)$:

$$In(t) \rightarrow Out(t + k) \tag{10}$$
$$P_Q(\mathbf{x}, t) \wedge Out(t) \rightarrow G(\mathbf{x}, t) \tag{11}$$
$$In(t) \rightarrow B(t + \ell) \tag{12}$$
$$P(\mathbf{x}, t) \wedge B(t) \rightarrow P'(\mathbf{x}, t) \tag{13}$$

We define $Q_1 = \langle G, \Pi^{d + \mathrm{rad}(Q)} \rangle$ and $Q_2 = \langle G, \Pi^s \rangle$. Intuitively, given a dataset for which In holds for a set of time points $\mathbf{T}$, query Q_1 captures the answers to Q at time points τ_{out} within the the interval $[\tau - d, \tau - s + \mathrm{rad}(Q)]$ for some $\tau \in \mathbf{T}$; in turn, for each such interval $[\tau - d, \tau - s + \mathrm{rad}(Q)]$, query Q_2 further restricts the answers to Q_1 to those that depend on input facts holding after $\tau - s$.

Since these queries can again be constructed in LOGSPACE, we obtain the desired upper bound.

Theorem 8. WINDOW *restricted to nonrecursive queries that are connected and constant-free is* co-NEXP-*complete.*

5 Related Work

The main challenges posed by stream processing and the basic architecture of a stream management system were first discussed in (Babu and Widom 2001; Babcock et al. 2002). Arasu, Babu, and Widom (2006) proposed the CQL query language, which extends SQL with a notion of *window*—a mechanism that allows one to reduce stream processing to traditional query evaluation. Since then, there have been numerous extensions and variants of CQL, which include a number of stream query languages for the Semantic Web (Barbieri et al. 2009; Le-Phuoc et al. 2011; 2013; Dell'Aglio et al. 2015).

In recent years, there have been several proposals for a general-purpose rule-based language in the context of stream reasoning. Streamlog (Zaniolo 2012) is a temporal Datalog language, which differs from the language considered in our paper in that it provides nonmonotonic negation and restricts the syntax so that only facts over time points explicitly present in the data can be derived. Furthermore, the focus in (Zaniolo 2012) is on dealing with so-called 'blocking queries', which are those whose answers may depend on input facts arbitrarily far in the future; for this, a syntactic fragment of the language is provided that precludes

blocking queries. LARS is a temporal rule-based language featuring window constructs and negation interpreted according to the stable model semantics (Beck et al. 2015; Beck, Dao-Tran, and Eiter 2015; Beck, Dao-Tran, and Eiter 2016). The semantics of LARS is rather different from that of temporal Datalog; in particular, the number of time points in a model is considered as part of the input to query evaluation, and hence is restricted to be finite; furthermore, the notion of window is built-in in LARS.

Stream reasoning has been studied in the context of RDF-Schema (Barbieri et al. 2010), and ontology-based data access (Calbimonte, Corcho, and Gray 2010; Özçep, Möller, and Neuenstadt 2014). In these works, the input data is assumed to arrive as a stream, but the ontology language is assumed to be nontemporal. Stream reasoning has also been considered in the unrelated context of complex event processing (Anicic et al. 2011; Dao-Tran and Le-Phuoc 2015).

There have been a number of proposals for rule-based languages in the context of temporal reasoning; here, the focus is on query evaluation over static temporal data, rather than on reasoning problems that are specific to stream processing. Our temporal Datalog language is a notational variant of Datalog$_{1S}$—the core language for temporal deductive databases (Chomicki and Imieliński 1988; Chomicki and Imieliński 1989; Chomicki 1990). Templog is an extension of Datalog with modal temporal operators (Abadi and Manna 1989), which was shown to be captured by Datalog$_{1S}$ (Baudinet, Chomicki, and Wolper 1993). Datalog was extended with integer periodicity and gap-order constraints in (Toman and Chomicki 1998); such constraints allow for the representation of infinite periodic phenomena. Finally, DatalogMTL is a recent Datalog extension based on metric temporal logic (Brandt et al. 2017).

In the setting of database constraint checking, a problem related to our window problem was considered by Chomicki (1995), who obtained some positive results for queries formulated in temporal first-order logic.

6 Conclusion and Future Work

In this paper, we have proposed novel decision problems relevant to the design of stream reasoning algorithms, and have studied their computational properties for temporal Datalog. These problems capture the key challenges behind rule-based stream reasoning, where rules can propagate information both to past and future time points. Our results suggest that rule-based stream reasoning is feasible in practice for nonrecursive temporal Datalog queries. Our problems are, however, either intractable in data complexity or undecidable in the general case.

We have made several mild technical assumptions in our upper bounds for nonrecursive queries, which we plan to lift in future work. Furthermore, we have assumed throughout the paper that numbers in the input are encoded in unary; we are currently looking into the impact of binary encoding on the complexity of our problems. Finally, we are planning to study extensions of nonrecursive temporal Datalog for which decidability of all our problems can be ensured.

Acknowledgments

References

Abadi, M., and Manna, Z. 1989. Temporal logic programming. *J. Symb. Comput.* 8(3):277–295.

Anicic, D.; Fodor, P.; Rudolph, S.; and Stojanovic, N. 2011. EP-SPARQL: a unified language for event processing and stream reasoning. In *WWW*, 635–644.

Arasu, A.; Babu, S.; and Widom, J. 2006. The CQL continuous query language: Semantic foundations and query execution. *VLDB J.* 15(2):121–142.

Babcock, B.; Babu, S.; Datar, M.; Motwani, R.; and Widom, J. 2002. Models and issues in data stream systems. In *PODS*, 1–16.

Babu, S., and Widom, J. 2001. Continuous queries over data streams. *SIGMOD Rec.* 30(3):109–120.

Barbieri, D. F.; Braga, D.; Ceri, S.; Della Valle, E.; and Grossniklaus, M. 2009. C-SPARQL: SPARQL for continuous querying. In *WWW*, 1061–1062.

Barbieri, D. F.; Braga, D.; Ceri, S.; Valle, E. D.; and Grossniklaus, M. 2010. Incremental reasoning on streams and rich background knowledge. In *ESWC*, 1–15.

Baudinet, M.; Chomicki, J.; and Wolper, P. 1993. Temporal deductive databases. In Tansel, A. U.; Clifford, J.; Gadia, S.; Jajodia, S.; Segev, A.; and Snodgrass, R., eds., *Temporal Databases*. Benjamin Cummings. 294–320.

Beck, H.; Dao-Tran, M.; Eiter, T.; and Fink, M. 2015. LARS: A logic-based framework for analyzing reasoning over streams. In *AAAI*, 1431–1438.

Beck, H.; Dao-Tran, M.; and Eiter, T. 2015. Answer update for rule-based stream reasoning. In *IJCAI*, 2741–2747.

Beck, H.; Dao-Tran, M.; and Eiter, T. 2016. Equivalent stream reasoning programs. In *IJCAI*, 929–935.

Benedikt, M., and Gottlob, G. 2010. The impact of virtual views on containment. *PVLDB* 3(1-2):297–308.

Brandt, S.; Kalayci, E. G.; Kontchakov, R.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2017. Ontology-based data access with a horn fragment of metric temporal logic. In *AAAI*, 1070–1076.

Calbimonte, J.-P.; Corcho, O.; and Gray, A. J. 2010. Enabling ontology-based access to streaming data sources. In *ISWC*, 96–111.

Chomicki, J., and Imieliński, T. 1988. Temporal deductive databases and infinite objects. In *PODS*, 61–73.

Chomicki, J., and Imieliński, T. 1989. Relational specifications of infinite query answers. In *SIGMOD*, 174–183.

Chomicki, J. 1990. Polynomial time query processing in temporal deductive databases. In *PODS*, 379–391.

Chomicki, J. 1995. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Trans. Database Syst.* 20(2):149–186.

Cosad, C.; Dufrene, K.; Heidenreich, K.; McMillon, M.; Jermieson, A.; O'Keefe, M.; and Simpson, L. 2009. Wellsite support from afar. *Oilfield Review* 21(2):48–58.

Dao-Tran, M., and Le-Phuoc, D. 2015. Towards enriching CQELS with complex event processing and path navigation. In *HiDeSt@KI*, 2–14.

Dao-Tran, M.; Beck, H.; and Eiter, T. 2015. Towards comparing RDF stream processing semantics. In *HiDeSt@KI*, 15–27.

Dell'Aglio, D.; Calbimonte, J.; Valle, E. D.; and Corcho, Ó. 2015. Towards a unified language for RDF stream query processing. In *ESWC (Satellite Events)*, 353–363.

Le-Phuoc, D.; Dao-Tran, M.; Parreira, J. X.; and Hauswirth, M. 2011. A native and adaptive approach for unified processing of linked streams and linked data. In *ISWC*, 370–388.

Le-Phuoc, D.; Quoc, H. N. M.; Le Van, C.; and Hauswirth, M. 2013. Elastic and scalable processing of linked stream data in the cloud. In *ISWC*, 280–297.

Münz, G., and Carle, G. 2007. Real-time analysis of flow data for network attack detection. In *IM*, 100–108.

Nuti, G.; Mirghaemi, M.; Treleaven, P.; and Yingsaeree, C. 2011. Algorithmic trading. *IEEE Computer* 44(11):61–69.

Özçep, Ö. L.; Möller, R.; and Neuenstadt, C. 2014. A stream-temporal query language for ontology based data access. In *KI*, 183–194.

Ronca, A.; Kaminski, M.; Cuenca Grau, B.; Motik, B.; and Horrocks, I. 2017. Stream reasoning in temporal Datalog. *CoRR* abs/1711.04013.

Shmueli, O. 1993. Equivalence of datalog queries is undecidable. *J. Log. Program.* 15(3):231–241.

Toman, D., and Chomicki, J. 1998. Datalog with integer periodicity constraints. *J. Log. Program.* 35(3):263–290.

Zaniolo, C. 2012. Logical foundations of continuous query languages for data streams. In *Datalog 2.0*, 177–189.

Definition 12. *A* derivation *of a fact α from a program Π is a finite labelled tree such that: (i) each node is labelled with a ground instance of a rule in Π; (ii) fact α is the head of the rule labelling the root; (iii) if the rule of a node w has a non-empty body containing atoms $\alpha_1, \ldots, \alpha_m$, then w has m children and α_i is the head of the rule labelling the i-th child.*

A Proofs for Section 4.1

Proposition 1. DTP *holds for I iff $\mathbf{o} \in Q(D \cup U, \tau_{out})$ implies $\mathbf{o} \in Q(D, \tau_{out})$ for every tuple $\mathbf{o}$ over Ω_I and every τ_{in}-update U involving only objects in Ω_I.*

Proof. If DTP holds for I, then the condition of the proposition clearly holds as well. For the converse, assume that DTP does not hold for I and hence there exists a tuple $\mathbf{c}$ of objects and a τ_{in}-update U such that $\mathbf{c} \in Q(D \cup U, \tau_{out})$ and $\mathbf{c} \notin Q(D, \tau_{out})$. Let h be the function mapping every object $o \in \Omega_I$ to itself and every other object to o_I, let $\mathbf{o} = h(\mathbf{c})$, and let $V = h(U)$. Clearly, $\mathbf{o} \in Q(D \cup V, \tau_{out})$, where $\mathbf{o}$ is a tuple over Ω_I, and V is a τ_{in}-update involving only objects in Ω_I. It remains to show that $\mathbf{o} \notin Q(D, \tau_{out})$. If $o_I \notin \mathbf{o}$, then $\mathbf{o} = \mathbf{c}$ and $\mathbf{c} \notin Q(D, \tau_{out})$ holds by our assumption. Otherwise, $\mathbf{o} \notin Q(D, \tau_{out})$ holds because o_I does not occur in $\Pi_Q \cup D$. $\qquad\square$

Lemma 1. DTP *holds for I iff $\mathbf{o} \in \Theta_I(D \cup \Upsilon_I, \tau_{out})$ implies $\mathbf{o} \in \Theta_I(D, \tau_{out})$ for every tuple $\mathbf{o}$ over Ω_I.*

Proof. Assume that DTP holds for I. We show that $\mathbf{o} \in \Theta_I(D \cup \Upsilon_I, \tau_{out})$ implies $\mathbf{o} \in \Theta_I(D, \tau_{out})$ for every tuple $\mathbf{o}$ over Ω_I. Let $\mathbf{o}$ be a tuple over Ω_I such that $\mathbf{o} \in \Theta_I(D \cup \Upsilon_I, \tau_{out})$. Let δ be a derivation of $P_Q(\mathbf{o}, \tau_{out})$ from $\Pi_{\Theta_I} \cup D \cup \Upsilon_I$. Let δ' be the derivation obtained from δ by first removing each node labelled by an instance of any of the additional rules introduced in Definition 6 and then replacing each P' with its corresponding EDB predicate P—i.e., the predicate P such that $P' = \psi(P)$. Let U be the τ_{in}-update consisting of each temporal EDB fact labelling a leaf of δ' and having time argument strictly bigger than τ_{in}. Then, by the construction of Θ_I, δ' is a derivation of $P_Q(\mathbf{o}, \tau_{out})$ from $\Pi_Q \cup D \cup U$, and hence $\mathbf{o} \in Q(D \cup U, \tau_{out})$. Therefore, $\mathbf{o} \in Q(D, \tau_{out})$ by Proposition 1 because DTP holds for I by assumption, and hence $\mathbf{o} \in \Theta_I(D, \tau_{out})$ by the previously observed properties of Θ_I.

For the converse, assume that $\mathbf{o} \in \Theta_I(D \cup \Upsilon_I, \tau_{out})$ implies $\mathbf{o} \in \Theta_I(D, \tau_{out})$ for every tuple $\mathbf{o}$ over Ω_I. We prove that DTP holds for I using Proposition 1, by showing that $\mathbf{o} \in Q(D \cup U, \tau_{out})$ implies $\mathbf{o} \in Q(D, \tau_{out})$ for every tuple $\mathbf{o}$ over Ω_I and τ_{in}-update U involving only objects of Ω_I. Let $\mathbf{o}$ be such a tuple and U such a τ_{in}-update, and suppose $\mathbf{o} \in Q(D \cup U, \tau_{out})$. By the previously observed properties of Θ_I and Υ_I, we then have $\mathbf{o} \in \Theta_I(D \cup \Upsilon_I, \tau_{out})$, and hence $\mathbf{o} \in \Theta_I(D, \tau_{out})$ by assumption. Therefore, $\mathbf{o} \in Q(D, \tau_{out})$ by the properties of Θ_I. $\qquad\square$

Lemma 6. *There exists a* LOGSPACE-*computable many-one reduction ϕ from query evaluation to* DTP *such that, for each instance $I = \langle Q, D, \mathbf{a} \rangle$ of query evaluation, the query Q' in $\phi(I)$ is independent of D and $\mathbf{a}$.*

Proof. Let $I = \langle Q, D, \mathbf{a} \rangle$ be an instance of query evaluation. We assume w.l.o.g. that Q is temporal and hence $\mathbf{a} = \langle \mathbf{o}, \tau \rangle$ with $\mathbf{o}$ a tuple of objects—otherwise, simply consider tuple $\langle \mathbf{a}, 0 \rangle$ instead of $\mathbf{a}$ and query $\langle P, \Pi \rangle$ with $\Pi = \Pi_Q \cup \{P_Q(\mathbf{x}) \to P(\mathbf{x}, 0)\}$ instead of Q.

We now define the instance $\phi(I)$ of DTP corresponding to I. Let T and A be fresh temporal predicates, where T is EDB and unary and A is EDB and of the same arity as P_Q. Let $D' = D \cup \{A(\mathbf{o}, \tau)\}$ and let Q' of the same arity as Q where $\Pi_{Q'}$ is Π_Q extended with the following rules:

$$T(t+1) \wedge A(\mathbf{x}, t) \to P_{Q'}(\mathbf{x}, t)$$
$$P_Q(\mathbf{x}, t) \wedge A(\mathbf{x}, t) \to P_{Q'}(\mathbf{x}, t)$$

We argue that $\mathbf{o} \in Q(D, \tau)$ if and only if DTP holds for $\phi(I) = \langle Q', D', \tau, \tau \rangle$. If $\mathbf{o} \in Q(D, \tau)$, we show that $Q'(D' \cup U, \tau) \subseteq Q'(D', \tau)$ for every τ-update U and hence DTP holds for $\phi(I)$. Assume that $\mathbf{c} \in Q'(D' \cup U, \tau)$ for some τ-update U. Then, since $P_{Q'}(\mathbf{c}, \tau)$ can only be entailed by one of the two new rules in $\Pi_{Q'}$, dataset $D' \cup U$ must contain the fact $A(\mathbf{c}, \tau)$; note, however, that this fact cannot be contained in U because U is a τ-update, and it is also not in D because it mentions A. Therefore, $\mathbf{c} = \mathbf{o}$; but now, by the assumption that $\mathbf{o} \in Q(D, \tau)$ and by the construction of Q' and D', we have $\mathbf{o} \in Q'(D', \tau)$, as required.

Next, assume that DTP holds for $\phi(I) = \langle Q', D', \tau, \tau \rangle$. We show that $\mathbf{o} \in Q(D, \tau)$. Consider the τ-update U containing the fact $T(\tau + 1)$. Then, $\mathbf{o} \in Q'(D' \cup U, \tau)$ because $A(\mathbf{o}, \tau) \in D'$, and hence $\mathbf{o} \in Q'(D', \tau)$ because DTP holds for $\phi(I)$ by assumption. We have that $\mathbf{o} \in Q'(D', \tau)$ implies $\mathbf{o} \in Q(D', \tau)$ because $T(\tau+1) \notin D'$ since T is fresh. Therefore, $\mathbf{o} \in Q(D, \tau)$ because $D' \setminus D = \{A(\mathbf{o}, \tau)\}$ and A does not occur in Q. $\qquad\square$

Theorem 1. DTP *is* PSPACE-complete *in data complexity.*

Proof. Hardness follows by Lemma 6, since query evaluation is PSPACE-complete in data complexity by the results in (Chomicki and Imieliński 1988).

We show an algorithm that decides DTP on $I = \langle Q, D, \tau_{in}, \tau_{out} \rangle$ in polynomial space if the query Q is considered fixed. According to Lemma 1, it is sufficient to iterate over all tuples $\mathbf{o}$ of objects from Ω_I, rejecting if $\mathbf{o} \in \Theta_I(D \cup \Upsilon_I, \tau_{out})$ and $\mathbf{o} \notin \Theta_I(D, \tau_{out})$, and accepting if we can complete all the iterations without rejecting. Let a be the maximum arity of a

predicate in Q, let c be the number of objects in $\Pi_Q \cup D$, and let p be the number of predicates in Q. Note that, with respect to the size of the input, a and p are constant and c is linear. We can build Θ_I in constant time because Θ_I depends only on Q, and we can build Υ_I in polynomial time because the number of facts in Υ_I is at most $1 + p \cdot (c+1)^a$. The number of iterations is polynomial because the number relevant object tuples is $(c+1)^a$. Finally, note that we can check both $\mathbf{o} \in \Theta_I(D \cup \Upsilon_I, \tau_{out})$ and $\mathbf{o} \notin \Theta_I(D \cup \Upsilon_I, \tau_{out})$ in polynomial space, since query evaluation and its complement are PSPACE-complete in data complexity by the results in (Chomicki and Imieliński 1988). $\qquad\square$

Nonrecursive case

Lemma 2. *Let a be the maximum radius of a rule in Π_Q and let $\mathbf{T}$ consist of τ_{out} and the time points in Π_Q. If $\Pi_Q \cup D \cup U \models P(\mathbf{o}, \tau)$ for a τ_{in}-update U involving only objects in Ω_I and a predicate P in Π_Q, and $|\tau - \tau'| \leq a \cdot (\mathrm{rank}(\Pi_Q) - \mathrm{rank}(P))$ for some $\tau' \in \mathbf{T}$, then $\Pi_Q \cup D \cup \Upsilon_I^{\mathrm{b}} \models P(\mathbf{o}, \tau)$.*

Proof. We prove the claim by induction on the rank of P. We assume w.l.o.g. that U contains only predicates in Π_Q.

In the base case $\mathrm{rank}(P) = 0$. Let $\Pi_Q \cup D \cup U \models P(\mathbf{o}, \tau)$ such that $|\tau - \tau'| \leq a \cdot (\mathrm{rank}(\Pi_Q) - \mathrm{rank}(P))$ for some $\tau' \in \mathbf{T}$. We show $\Pi_Q \cup D \cup \Upsilon_I^{\mathrm{b}} \models P(\mathbf{o}, \tau)$. Since $\mathrm{rank}(P) = 0$, we have $P(\mathbf{o}, \tau) \in \Pi_Q \cup D \cup U$ because P occurs only in facts. Moreover, we have $|\tau - \tau'| \leq a \cdot \mathrm{rank}(\Pi_Q) \leq \mathrm{rad}(\Pi_Q)$. We distinguish two cases. If $\tau \leq \tau_{in}$, then $P(\mathbf{o}, \tau) \in \Pi_Q \cup D$ since U only contains facts with time points after τ_{in}, and the claim follows. Otherwise, we have $\tau_{in} < \tau \leq \tau' + \mathrm{rad}(\Pi_Q)$, and hence τ is critical. Since Υ_I^{b} contains all facts involving only EDB predicates in Π_Q, objects in Ω_I, and critical time points, we then have $P(\mathbf{o}, \tau) \in \Upsilon_I^{\mathrm{b}}$, and the claim follows.

For the inductive step, we assume that the claim holds for every predicate of rank at most n and show it for $\mathrm{rank}(P) = n+1$. Let $\Pi_Q \cup D \cup U \models P(\mathbf{o}, \tau)$ such that $|\tau - \tau'| \leq a \cdot (\mathrm{rank}(\Pi_Q) - n - 1)$ for some $\tau' \in \mathbf{T}$. Let δ be a derivation of $P(\mathbf{o}, \tau)$ from $\Pi_Q \cup D \cup U$. We show $\Pi_Q \cup D \cup \Upsilon_I^{\mathrm{b}} \models P(\mathbf{o}, \tau)$. Let r be the label of the root of δ, and let r' be a rule in Π_Q such that r is an instance of r'. It suffices to show that $\Pi_Q \cup D \cup \Upsilon_I^{\mathrm{b}} \models \alpha$ for each atom $\alpha \in \mathrm{body}(r)$. Let α be an arbitrary such atom. We have $\alpha = P_1(\mathbf{o}_1, \tau_1)$ for τ_1 a time term, by our assumption that Q does not contain rigid atoms. Since δ is a derivation, we have $\Pi_Q \cup D \cup U \models \alpha$. We distinguish two subcases.

If the atom corresponding to α in r' mentions a time variable, so does its head because Π_Q is connected, and hence we have $|\tau_1 - \tau| \leq a$. Consequently, since $|\tau - \tau'| \leq a \cdot (\mathrm{rank}(\Pi_Q) - n - 1)$, we have $|\tau_1 - \tau'| \leq a \cdot (\mathrm{rank}(\Pi_Q) - n) \leq a \cdot (\mathrm{rank}(\Pi_Q) - \mathrm{rank}(P_1))$; $\Pi_Q \cup D \cup \Upsilon_I^{\mathrm{b}} \models \alpha$ then follows from $\Pi_Q \cup D \cup U \models \alpha$ by the inductive hypothesis.

If the atom corresponding to α in r' mentions no time variable, τ_1 must be a time point, and hence $\tau_1 \in \mathbf{T}$. Clearly, $|\tau_1 - \tau_1| = 0 \leq a \cdot (\mathrm{rank}(\Pi_Q) - n)$, and hence $\Pi_Q \cup D \cup \Upsilon_I^{\mathrm{b}} \models \alpha$ follows from $\Pi_Q \cup D \cup U \models \alpha$ by the inductive hypothesis. $\qquad\square$

Lemma 3. DTP *holds for I iff $\mathbf{o} \in Q(D \cup \Upsilon_I^{\mathrm{b}}, \tau_{out})$ implies $\mathbf{o} \in Q(D, \tau_{out})$ for every tuple $\mathbf{o}$ over Ω_I.*

Proof. If DTP holds for I, then trivially $\mathbf{o} \in Q(D \cup \Upsilon_I^{\mathrm{b}}, \tau_{out})$ implies $\mathbf{o} \in Q(D, \tau_{out})$ for every tuple $\mathbf{o}$ over Ω_I, because Υ_I^{b} is a τ_{in}-update. For the converse, assume that $\mathbf{o} \in Q(D \cup \Upsilon_I^{\mathrm{b}}, \tau_{out})$ implies $\mathbf{o} \in Q(D, \tau_{out})$ for every tuple $\mathbf{o}$ over Ω_I. We prove that DTP holds for I by showing that $\mathbf{o} \in Q(D \cup U, \tau_{out})$ implies $\mathbf{o} \in Q(D, \tau_{out})$ for every tuple $\mathbf{o}$ over Ω_I and τ_{in}-update U involving only objects of Ω_I; the claim then holds by Proposition 1. Let $\mathbf{o}$ be a tuple over Ω_I and let U be a τ_{in}-update involving only objects of Ω_I such that $\mathbf{o} \in Q(D \cup U, \tau_{out})$. Since $\tau_{out} \in \mathbf{T}$ and $|\tau_{out} - \tau_{out}| = 0$, by Lemma 2 for $\tau = \tau' = \tau_{out}$ it then follows that $\mathbf{o} \in Q(D \cup \Upsilon_I, \tau_{out})$. By our assumption, $\mathbf{o} \in Q(D, \tau_{out})$. $\qquad\square$

Theorem 2. DTP *is in* P *in data complexity if restricted to nonrecursive connected queries.*

Proof. We show an algorithm that decides DTP on $I = \langle Q, D, \tau_{in}, \tau_{out} \rangle$ in polynomial time if the query Q is considered fixed. According to Lemma 3, it is sufficient to iterate over all tuples $\mathbf{o}$ of objects from Ω_I, rejecting if $\mathbf{o} \in Q(D \cup \Upsilon_I^{\mathrm{b}}, \tau_{out})$ and $\mathbf{o} \notin Q(D, \tau_{out})$, and accepting if we can complete all the iterations without rejecting. Let a be the maximum arity of a predicate in Q, let c be the number of objects in $\Pi_Q \cup D$, and let p be the number of predicates in Q. Note that, with respect to the size of the input, the values a, p and $\mathrm{rad}(Q)$ are constant; furthermore, $\tau_0 - \tau_{in}$ and c are linear. We can build Υ_I^{b} in polynomial time because the number of facts in Υ_I^{b} is bounded by $p \cdot (\mathrm{rad}(Q) + \tau_0 - \tau_{in}) \cdot (c+1)^a$. The number of iterations is polynomial because the number of relevant object tuples is c^a. Finally, checking both $\mathbf{o} \in Q(D \cup \Upsilon_I^{\mathrm{b}}, \tau_{out})$ and $\mathbf{o} \notin Q(D, \tau_{out})$ is in AC^0. $\qquad\square$

B   Proofs for Section 4.2

Lemma 7. *There exists a* LOGSPACE-*computable many-one reduction ϕ from datalog query containment to* FORGET *such that, for every instance $I = \langle Q_1, Q_2 \rangle$ of datalog query containment, the query in $\phi(I)$ is nonrecursive if Q_1 and Q_2 are nonrecursive.*

Proof. Let $\langle Q_1, Q_2 \rangle$ be an instance of query containment with Q_1 and Q_2 datalog queries. Without loss of generality, $P_{Q_1} = P_{Q_2} = G$. For any rigid n-ary IDB predicate P and $i \in \{1, 2\}$, let P_i be a fresh rigid n-ary IDB predicate uniquely associated with P and i. For any rigid n-ary EDB (resp., IDB) predicate P, let P^t be a fresh temporal $(n + 1)$-ary EDB (IDB) predicate uniquely associated with P. Let t be a time variable. For $i \in \{1, 2\}$, let Π_i be Π_{Q_i} after replacing each rigid n-ary IDB predicate P with P_i; let Π'_i be Π_i after replacing each rigid atom $P(\mathbf{u})$ with the temporal atom $P^t(\mathbf{u}, t)$. Let A be a fresh temporal unary EDB predicate. Let Q be the query such that P_Q is a fresh temporal IDB predicate of the same arity as G_1^t (or, equivalently, as G_2^t), and Π_Q is $\Pi'_1 \cup \Pi'_2$ extended with the following rules:

$$A(t - 1) \wedge G_1^t(\mathbf{x}, t) \rightarrow P_Q(\mathbf{x}, t) \tag{14}$$

$$G_2^t(\mathbf{x}, t) \rightarrow P_Q(\mathbf{x}, t) \tag{15}$$

Clearly, Q can be constructed in logarithmic space w.r.t. the size of Q_1 and Q_2.

Let $D = \{A(0)\}$. We show that $Q_1 \sqsubseteq Q_2$ iff FORGET holds for $\phi(\langle Q_1, Q_2 \rangle) = \langle Q, D, 1, 1, 0 \rangle$.

Assume that $Q_1 \sqsubseteq Q_2$ holds. We show that FORGET holds for $\langle Q, D, 1, 1, 0 \rangle$, by showing that $Q(D \cup U, \tau) \subseteq Q(D[0] \cup U, \tau)$ for every 1-update U and time point $\tau \geq 1$. Let $\mathbf{o}$ be a tuple of objects, let U be a 1-update and let $\tau \geq 1$ such that $\mathbf{o} \in Q(D \cup U, \tau)$. Let δ be a derivation of $P_Q(\mathbf{o}, \tau)$ from $\Pi_Q \cup D \cup U$. The root of δ is labelled with an instance of either rule (14) or rule (15) since P_Q does not occur in $\Pi'_1 \cup \Pi'_2$. We consider the two cases separately. If the label is an instance of rule (14), then $\Pi_Q \cup D \cup U \models G_1^t(\mathbf{o}, \tau)$; then $\Pi'_1 \cup D \cup U \models G_1^t(\mathbf{o}, \tau)$ by the construction of Π_Q; then $\Pi'_1 \cup U \models G_1^t(\mathbf{o}, \tau)$ because facts in Π'_1 all have t as time argument, and τ satisfies $\tau \geq 1$ by assumption; then $\mathbf{o} \in Q_1(U')$ by the construction of Π'_1, where U' is the dataset consisting of each rigid fact $P(\mathbf{c})$ for $P^t(\mathbf{c}, \tau) \in U$; then $\mathbf{o} \in Q_2(U')$ since $Q_1 \sqsubseteq Q_2$ by assumption; then $\Pi'_2 \cup U \models G_2^t(\mathbf{o}, \tau)$ by the construction of Π'_2; then $\Pi_Q \cup U \models G_2^t(\mathbf{o}, \tau)$ because $\Pi'_2 \subseteq \Pi_Q$; then $\mathbf{o} \in Q(U, \tau)$ by rule (15), and hence $\mathbf{o} \in Q(D[0] \cup U, \tau)$ since $D[0] = \emptyset$. If the root of δ is labelled with an instance of rule (15), then $\Pi_Q \cup D \cup U \models G_2^t(\mathbf{o}, \tau)$; then $\Pi'_2 \cup D \cup U \models G_2^t(\mathbf{o}, \tau)$ by the construction of Π_Q; then $\Pi'_2 \cup U \models G_2^t(\mathbf{o}, \tau)$ because facts in Π'_2 all have t as time argument and τ satisfies $\tau \geq 1$ by assumption; then $\Pi_Q \cup U \models G_2^t(\mathbf{o}, \tau)$ because $\Pi'_2 \subseteq \Pi_Q$, and so $\Pi_Q \cup D[0] \cup U \models G_2^t(\mathbf{o}, \tau)$ because $D[0] = \emptyset$.

For the converse, assume that FORGET holds for $\langle Q, D, 1, 1, 0 \rangle$, and hence $Q(D \cup U, \tau) \subseteq Q(D[0] \cup U, \tau)$ for every 1-update U and time point $\tau \geq 1$. We show $Q_1(D') \subseteq Q_2(D')$ for every dataset D'. Let $\mathbf{o}$ be a tuple of objects and D' a dataset such that $\mathbf{o} \in Q_1(D')$. Let U be the 1-update consisting of each fact $P^t(\mathbf{c}, 2)$ for $P(\mathbf{c}) \in D'$. Then, $\Pi'_1 \cup U \models G_1^t(\mathbf{o}, 2)$ by the construction of Π'_1; then $\Pi_Q \cup U \models G_1^t(\mathbf{o}, 2)$ because $\Pi'_1 \subseteq \Pi_Q$, and hence $\mathbf{o} \in Q(D \cup U, 2)$ by rule (14). It follows that $\mathbf{o} \in Q(D[0] \cup U, 2)$ by assumption, and hence $\mathbf{o} \in Q(U, 2)$ because $D[0] = \emptyset$. Then the root of every derivation of $P_Q(\mathbf{o}, 2)$ from $\Pi_Q \cup U$ must be an instance of rule (15). Therefore, $\Pi_Q \cup U \models G_2^t(\mathbf{o}, 2)$; then $\Pi'_2 \cup U \models G_2^t(\mathbf{o}, 2)$ by the construction of Π_Q, and hence $\mathbf{o} \in Q_2(D')$ by the construction of Π'_2 and U. $\qquad\square$

Theorem 3. FORGET *is undecidable.*

Proof. The claim follows by Lemma 7 since query containment for datalog is undecidable by the results in (Shmueli 1993). $\qquad\square$

Nonrecursive Case

Lemma 8. *Let a be the maximum radius of a rule in Π_Q. For each dataset D', predicate P, objects $\mathbf{o}$, time point τ and set $\mathbf{T}$ containing τ and each time point in Π_Q, $\Pi_Q \cup D' \models P(\mathbf{o}, \tau)$ implies $\Pi_Q \cup D'[\min(\mathbf{T}) - a \cdot \mathrm{rank}(P) - 1] \models P(\mathbf{o}, \tau)$.*

Proof. We proceed by induction on $\mathrm{rank}(P)$. For the base case, let $\mathrm{rank}(P) = 0$, and suppose $\Pi_Q \cup D' \models P(\mathbf{o}, \tau)$ for some D', $\mathbf{o}$ and τ. Since $\mathrm{rank}(P) = 0$, P must be EDB (otherwise, facts involving P cannot be entailed by $\Pi_Q \cup D'$ as P does not occur in rule heads in Π_Q and may not occur in D'), and hence $P(\mathbf{o}, \tau) \in \Pi_Q \cup D'$. But then $P(\mathbf{o}, \tau) \in \Pi_Q \cup D'[\min(\mathbf{T}) - 1]$ since $\min(\mathbf{T}) \leq \tau$; consequently, $\Pi_Q \cup D'[\min(\mathbf{T}) - 1] \models P(\mathbf{o}, \tau)$, as required.

For the inductive step, suppose the claim holds for all predicates of rank at most n, and let $\Pi_Q \cup D' \models P(\mathbf{o}, \tau)$ where $\mathrm{rank}(P) = n + 1$. We show $\Pi_Q \cup D'[\min(\mathbf{T}) - a \cdot (n + 1) - 1] \models P(\mathbf{o}, \tau)$. Let δ be a derivation of $P(\mathbf{o}, \tau)$ from $\Pi_Q \cup D'$ whose root is labelled with an instance r of a rule in Π_Q. Then, for each temporal body atom $\alpha = P'(\mathbf{o}', \tau')$ of r, we have $\Pi_Q \cup D' \models \alpha$. Hence, by the inductive hypothesis, for each such α we have $\Pi_Q \cup D'[\min(\mathbf{T}_\alpha) - a \cdot n_\alpha - 1] \models \alpha$, where $\mathbf{T}_\alpha$ is the set consisting of τ' and each time point in Π_Q, and $n_\alpha = \mathrm{rank}(P') \leq n$. Note that τ' is either a time point in $\mathbf{T}$ or $\tau' \geq \tau - a$; thus, $\min(\mathbf{T}_\alpha) \geq \min(\mathbf{T}) - a$, and hence $\min(\mathbf{T}_\alpha) - a \cdot n_\alpha - 1 \geq \min(\mathbf{T}) - a \cdot (n_\alpha + 1) - 1 \geq \min(\mathbf{T}) - a \cdot (n + 1) - 1$, where the last inequality holds since $n_\alpha \leq n$. Consequently, for each α, $D'[\min(\mathbf{T}_\alpha) - a \cdot n_\alpha - 1] \subseteq D'[\min(\mathbf{T}) - a \cdot (n + 1) - 1]$, and hence $\Pi_Q \cup D'[\min(\mathbf{T}) - a \cdot (n + 1) - 1] \models \alpha$ by monotonicity of entailment. On the other hand, for all rigid body atoms β in r, $\Pi_Q \cup D' \models \beta$ implies $\Pi_Q \cup D'[\min(\mathbf{T}) - a \cdot (n + 1) - 1] \models \beta$ since Π_Q is connected and hence the validity of β does not depend on temporal facts. The claim then follows by r. $\qquad\square$

Lemma 9. *Let a be the maximum radius of a rule in Π_Q and let $\mathbf{T}$ be the set of time points in Π_Q. For each time point τ_a, each dataset D', and each time point $\tau \leq \tau_0 + a \cdot (\mathrm{rank}(\Pi_Q) - \mathrm{rank}(P))$, where τ_0 is the maximum among τ_a and the time points in $\mathbf{T}$, $\Pi_Q \cup D' \models P(\mathbf{o}, \tau)$ implies $\Pi_Q \cup D'' \models P(\mathbf{o}, \tau)$, where D'' consists of each fact in D' with time argument τ' satisfying $\tau' \leq \tau_0 + \mathrm{rad}(\Pi_Q)$.*

Proof. Assume $\Pi_Q \cup D' \models P(\mathbf{o}, \tau)$ for each $\tau \leq \tau_0 + a \cdot (\text{rank}(\Pi_Q) - \text{rank}(P))$. We prove $\Pi_Q \cup D'' \models P(\mathbf{o}, \tau)$ by induction on the rank of P.

In the base case, $\text{rank}(P) = 0$. Since P occurs only in facts, $P(\mathbf{o}, \tau) \in \Pi_Q \cup D'$. If $\tau \in \mathbf{T}$, then $P(\mathbf{o}, \tau) \in \Pi_Q$, and hence $P(\mathbf{o}, \tau) \in \Pi_Q \cup D''$. Otherwise, $P(\mathbf{o}, \tau) \in D'$ and $\tau \leq \tau_0 + a \cdot \text{rank}(\Pi_Q) \leq \tau_0 + \text{rad}(\Pi_Q)$, and hence $P(\mathbf{o}, \tau) \in \Pi_Q \cup D''$ by the definition of D''. In either case, the claim follows.

For the inductive step, we assume that the claim holds for every predicate of rank at most n and we show it for $\text{rank}(P) = n + 1$. Let δ be a derivation of $P(\mathbf{o}, \tau)$ from $\Pi_Q \cup D'$, let r label the root of δ, and let r' be a rule in Π_Q such that r is an instance of r'. It suffices to show $\Pi_Q \cup D'' \models \alpha$ for each atom $\alpha \in \text{body}(r)$. Let α be an arbitrary such atom. Since δ is a derivation, $\Pi_Q \cup D' \models \alpha$. We distinguish two cases.

If α is rigid, the claim follows from $\Pi_Q \cup D' \models \alpha$ since D' and D'' coincide on rigid facts and the validity of α depends only on rigid facts because Π_Q is connected.

Otherwise, we have $\alpha = P_1(\mathbf{o}_1, \tau_1)$. If $\tau_1 \in \mathbf{T}$, then $\tau_1 \leq \tau_0 \leq \tau_0 + a \cdot (\text{rank}(\Pi_Q) - \text{rank}(P_1))$; then $\Pi_Q \cup D'' \models \alpha$ by the inductive hypothesis. Now, let $\tau_1 \notin \mathbf{T}$. Then the atom corresponding to α in r' mentions a time variable t and $\text{head}(r')$ mentions the same variable t because r' is connected; hence, we have $|\tau_1 - \tau| \leq a$, and thus $\tau_1 \leq \tau_0 + a \cdot (\text{rank}(\Pi_Q) - n) \leq \tau_0 + a \cdot (\text{rank}(\Pi_Q) - \text{rank}(P_1))$. Therefore, $\Pi_Q \cup D'' \models \alpha$ by the inductive hypothesis. $\square$

Lemma 4. FORGET *holds for* I *iff* $Q_1 \sqsubseteq Q_2$, *where* Q_1 *and* Q_2 *are as given in Definition 11.*

Proof sketch. Assume $Q_1 \sqsubseteq Q_2$. We prove that FORGET holds for I by showing that $Q(D \cup U, \tau) \subseteq Q(D[\tau_{mem}] \cup U, \tau)$ for every τ_{in}-update U and time point $\tau \geq \tau_{out}$. Note that Q is connected and constant-free and, hence, so are Q_1 and Q_2. We argue that it suffices to show $Q(D \cup U, \tau) \subseteq Q(D[\tau_{mem}] \cup U, \tau)$ for every τ_{in}-update U and time point τ satisfying $\tau_{out} \leq \tau \leq \tau_{mem} + \text{rad}(Q)$. If $\mathbf{o} \in Q(D \cup U, \tau)$ for $\tau > \tau_{mem} + \text{rad}(Q)$, then $\mathbf{o} \in Q(D[\tau - \text{rad}(Q) - 1] \cup U, \tau)$ by Lemma 8, and hence $\mathbf{o} \in Q(D[\tau_{mem}] \cup U, \tau)$. Furthermore, by Lemma 9, it suffices to show that $\mathbf{o} \in Q(D \cup U, \tau)$ implies $\mathbf{o} \in Q(D[\tau_{mem}] \cup U, \tau)$ for every tuple $\mathbf{o}$, τ_{in}-update U with time points smaller than or equal to $\tau_{mem} + 2 \cdot \text{rad}(Q)$, and time point τ satisfying $\tau_{out} \leq \tau \leq \tau_{mem} + \text{rad}(Q)$. Now, let $\mathbf{o} \in Q(D \cup U, \tau)$ for U a τ_{in}-update with time points smaller than or equal to $\tau_{mem} + 2 \cdot \text{rad}(Q)$, and τ satisfying $\tau_{out} \leq \tau \leq \tau_{mem} + \text{rad}(Q)$. We have that $\mathbf{o} \in Q_1(U, \tau)$ by construction of Q_1. Hence $\mathbf{o} \in Q_2(U, \tau)$ by our assumption, and hence $\mathbf{o} \in Q(D[\tau_{mem}] \cup U, \tau)$ by construction of Q_2, as required.

For the converse, assume $Q_1 \not\sqsubseteq Q_2$. There is a time point τ, dataset U and a tuple $\mathbf{o}$ such that $\mathbf{o} \in Q_1(U, \tau)$ and $\mathbf{o} \notin Q_2(U, \tau)$. In particular, $\mathbf{o} \in Q_1(U', \tau)$ holds for the subset of U containing time points τ' with $\tau_{in} < \tau' \leq \tau_{mem} + 2 \cdot \text{rad}(Q)$ by construction of Q_1. Note that U' is a τ_{in}-update. Furthermore, τ satisfies $\tau_{out} \leq \tau \leq \tau_{mem} + \text{rad}(Q)$ by construction of Q_1. Then, $\mathbf{o} \in Q_1(U', \tau)$ implies $\mathbf{o} \in Q(D \cup U', \tau)$, and $\mathbf{o} \notin Q_2(U, \tau)$ implies $\mathbf{o} \notin Q_2(U', \tau)$ by monotonicity of entailment, which implies $\mathbf{o} \notin Q(D[\tau_{mem}] \cup U', \tau)$. Therefore, FORGET does not hold for I. $\square$

Theorem 4. FORGET *is in* P *in data complexity if restricted to nonrecursive connected queries whose rules contain no time points.*

Proof sketch. Let Q_1 and Q_2 be the left and right critical queries for I. In order to check whether FORGET holds for I, it suffices to check $Q_1 \sqsubseteq Q_2$ by Lemma 4. Clearly, Q_1 and Q_2 can be built in polynomial time. We argue next that $Q_1 \sqsubseteq Q_2$ can be checked in polynomial time.

Let E_i^τ, for a time point τ, be the (temporal) UCQ consisting of (the leaves of) each maximal unfolding of Q_i starting with the atom $P_Q(\mathbf{x}, \tau)$, and let $E_i = \bigvee_{\tau_{out} \leq \tau \leq \tau_{mem} + \text{rad}(Q)} E_i^\tau$. Clearly, E_i is equivalent to Q_i since, by construction, Q_i can only derive facts about P_Q between τ_{out} and $\tau_{mem} + \text{rad}(Q)$. Note that if Q is fixed, then for each E_i^τ, the number of conjuncts in each CQ in E_i^τ is bounded by a constant c, the arity of each such conjunct is bounded by a constant a, and the number of CQs in each E_i^τ is bounded by $r^{\text{rank}(\Pi_Q)+1}$, where r is the number of rules in Π_i'; importantly, r and hence $r^{\text{rank}(\Pi_Q)+1}$ is polynomial in the size of the input. Moreover, since Q is connected, E_i mentions no time variables, and hence each temporal CQ in E_i can be equivalently seen as a nontemporal CQ.

In order to check the containment $Q_1 \sqsubseteq Q_2$, we can equivalently check whether for each CQ Q_1' in E_1 there is a CQ Q_2' in E_2 such that $Q_1' \sqsubseteq Q_2'$; for this, it is well-known that we can equivalently check whether there is a containment mapping from Q_2' to Q_1'. The number of pairs of queries to check is bounded by $(\tau_{mem} + \text{rad}(Q) - \tau_{out} + 1) \cdot r^{2(\text{rank}(\Pi_Q)+1)}$. Furthermore, the size and number of possible containment mappings is bounded (resp., polynomially and exponentially) in $c \cdot a$. Hence, we can generate all the possible containment mappings for a pair of queries in constant time, and check them in polynomial time. $\square$

C Containment of nonrecursive queries

Lemma 5. *Query containment restricted to nonrecursive queries that are connected and constant-free is in* co-NEXP.

Proof sketch. We proceed by a reduction to datalog query containment. Without loss of generality, $P_{Q_1} = P_{Q_2} = G$ for some predicate G.

If G is rigid, since we have assumed that Q_1 and Q_2 contain no time points and are connected, G recursively depends only on rigid predicates. Therefore, it suffices to consider the containment problem for the nonrecursive datalog subprograms of Π_{Q_1} and Π_{Q_2}, which is co-NEXP-complete by the results in (Benedikt and Gottlob 2010).

If G is temporal, let m be the maximum between the radiuses of Π_{Q_1} and Π_{Q_2}, and let $\mathbf{T}$ be the time interval $[0, 2m]$. Let Π_i be the grounding of Π_{Q_i} on the temporal arguments with time points in $\mathbf{T}$. Note that, since we have assumed that Q_1 and Q_2 are connected, the size of Π_i is bounded by m times the size of Π_{Q_i}, i.e., cubically in the size of Π_{Q_i}. Let G' be a fresh temporal IDB predicate of the same arity as G. Let Π_i' be Π_i extended with the rule

$$G(\mathbf{x}, m) \rightarrow G'(\mathbf{x}, m)$$

Let Q_i' be the query $\langle G', \Pi_i' \rangle$. We have that $Q_1 \sqsubseteq Q_2$ iff $Q_1' \sqsubseteq Q_2'$, because *(i)* derivations of facts at m involve only time points in $\mathbf{T}$ and, *(ii)* for each dataset D and time point n, there is a dataset D' such that all derivations of facts at n w.r.t. $Q_1 \cup D$ and $Q_2 \cup D$ are isomorphic to derivations of facts at m w.r.t. $Q_1 \cup D'$ and $Q_2 \cup D'$. Finally, each Q_i' is temporally ground, and hence can be seen as a datalog query. The claim once again follows by (Benedikt and Gottlob 2010) since Q_1', Q_2' are polynomial in Q_1, Q_2. $\qquad\square$

D Proofs for Section 4.3

Delay

Lemma 10. *There exists a* LOGSPACE*-computable many-one reduction ϕ from containment of datalog queries to* DELAY *such that, for every instance $I = \langle Q_1, Q_2 \rangle$ of query containment, the query in $\phi(I)$ is nonrecursive if Q_1 and Q_2 are nonrecursive.*

Proof. Let $\langle Q_1, Q_2 \rangle$ be an instance of query containment with Q_1 and Q_2 datalog queries. Without loss of generality, $P_{Q_1} = P_{Q_2} = G$. For any rigid n-ary IDB predicate P and for $i \in \{1, 2\}$, let P_i be a fresh rigid n-ary IDB predicate uniquely associated with P and i. For any rigid n-ary EDB (resp., IDB) predicate P, let P^{t} be a fresh temporal $(n + 1)$-ary EDB (IDB) predicate uniquely associated with P. Let t be a time variable. For $i \in \{1, 2\}$, let Π_i be Π_{Q_i} after replacing each rigid n-ary IDB predicate P with P_i; and let Π_i' be Π_i after replacing each rigid atom $P(\mathbf{u})$ with the temporal atom $P^{\mathrm{t}}(\mathbf{u}, t)$. Let A be a fresh temporal unary EDB predicate. Let Q be the query such that P_Q is a fresh temporal IDB predicate of the same arity as G_i^{t}, and Π_Q is $\Pi_1' \cup \Pi_2'$ extended with the following rules:

$$A(t+1) \wedge G_1^{\mathrm{t}}(\mathbf{x}, t) \rightarrow P_Q(\mathbf{x}, t) \tag{16}$$

$$G_2^{\mathrm{t}}(\mathbf{x}, t) \rightarrow P_Q(\mathbf{x}, t) \tag{17}$$

It is easily seen that Q can be constructed in logarithmic space w.r.t. the size of Q_1 and Q_2.

We show that $Q_1 \sqsubseteq Q_2$ if and only if DELAY holds for $\phi(\langle Q_1, Q_2 \rangle) = \langle Q, 0 \rangle$.

Assume that $Q_1 \sqsubseteq Q_2$ holds. We prove that DELAY$(Q, 0)$ is true by showing $Q(D \cup U, \tau) \subseteq Q(D, \tau)$ for every dataset D, time point τ, and τ-update U. Let $\mathbf{o}$ be a tuple of objects, D a dataset, τ a time point, and let U be a τ-update such that $\mathbf{o} \in Q(D \cup U, \tau)$. Let δ be a derivation of $P_Q(\mathbf{o}, \tau)$ from $\Pi_Q \cup D \cup U$. Then the root of δ is labelled with an instance of either rule (16) or rule (17), since P_Q does not occur in $\Pi_1' \cup \Pi_2'$. First, suppose the root is labelled with an instance of rule (16). Clearly, $\Pi_Q \cup D \cup U \models G_1^{\mathrm{t}}(\mathbf{o}, \tau)$. Since the atoms of all rules in Π_Q but (16) have t as a time argument, no derivation δ' of $G_1^{\mathrm{t}}(\mathbf{o}, \tau)$ from $\Pi_Q \cup D \cup U$ contains a time point different from τ, and hence no atom in U; it follows that δ' is also a derivation of $G_1^{\mathrm{t}}(\mathbf{o}, \tau)$ from $\Pi_Q \cup D$, and hence $\Pi_Q \cup D \models G_1^{\mathrm{t}}(\mathbf{o}, \tau)$, which implies $\mathbf{o} \in Q_1(D')$ by construction of Q, where D' consists of each fact $P(\mathbf{c})$ for $P(\mathbf{c}, \tau) \in D$. Therefore, since $Q_1 \sqsubseteq Q_2$ by assumption, $\mathbf{o} \in Q_2(D')$. Thus, $\Pi_Q \cup D \models G_2^{\mathrm{t}}(\mathbf{o}, \tau)$ by the construction of Q, and hence $\mathbf{o} \in Q(D, \tau)$ by rule (17). Now, suppose the root of δ is labelled with an instance of rule (17). Clearly, $\Pi_Q \cup D \cup U \models G_2^{\mathrm{t}}(\mathbf{o}, \tau)$. Since the atoms of all rules in Π_Q but (16) have t as a time argument, no derivation δ' of $G_2^{\mathrm{t}}(\mathbf{o}, \tau)$ from $\Pi_Q \cup D \cup U$ contains a time point different from τ, and hence no atom in U; it follows that δ' is also a derivation of $G_2^{\mathrm{t}}(\mathbf{o}, \tau)$ from $\Pi_Q \cup D$, and hence $\Pi_Q \cup D \models G_2^{\mathrm{t}}(\mathbf{o}, \tau)$. Thus, $\mathbf{o} \in Q(D, \tau)$ by rule (17).

For the converse, assume that DELAY$(Q, 0)$ is true. We prove $Q_1 \sqsubseteq Q_2$ by showing $Q_1(D) \subseteq Q_2(D)$ for every dataset D. Let $\mathbf{o}$ be a tuple of objects and D a dataset such that $\mathbf{o} \in Q_1(D)$. By construction, $\Pi_Q \cup D' \models G_1^{\mathrm{t}}(\mathbf{o}, \tau)$ for each time point τ, where D' consists of each fact $P(\mathbf{c}, \tau)$ for $P(\mathbf{c}) \in D$. By rule (16), $\mathbf{o} \in Q(D' \cup U, \tau)$, where U is the τ-update containing $A(\tau + 1)$. Since DELAY$(Q, 0)$ is true by assumption, we have that $\mathbf{o} \in Q(D', \tau)$. Since $A(\tau + 1) \notin D'$ and P_Q does not occur in $\Pi_1' \cup \Pi_2'$, the root of any derivation of $P_Q(\mathbf{o}, \tau)$ from $\Pi_Q \cup D'$ must be labelled with an instance of rule (17), and hence $\Pi_Q \cup D' \models G_2^{\mathrm{t}}(\mathbf{o}, \tau)$. Therfore, $\mathbf{o} \in Q_2(D)$ by construction. $\qquad\square$

Theorem 5. DELAY *is undecidable.*

Proof. The claim follows by Lemma 10 since query containment for datalog is undecidable by the results in (Shmueli 1993). $\qquad\square$

Theorem 7. DELAY *restricted to nonrecursive queries that are connected and constant-free is* co-NEXP-complete.

Proof. We first prove hardness. The reduction ϕ in Lemma 10 is such that, for each instance $I = \langle Q_1, Q_2 \rangle$ of query containment, the query in $\phi(I)$ is connected, and is also nonrecursive constant-free if Q_1 and Q_2 are nonrecursive constant-free. Furthermore, query containment is co-NEXP-hard already for nonrecursive constant-free datalog queries, by the results in (Benedikt and Gottlob 2010).

For the upper bound, we show that there is a LOGSPACE-computable many-one reduction ϕ from DELAY restricted to non-recursive queries to query containment for nonrecursive queries; the result then follows by Lemma 5. Consider the construction given in Section 4.3. We show that $Q_1 \sqsubseteq Q_2$ iff DELAY(Q, d) holds.

Assume $Q_1 \sqsubseteq Q_2$. We show that DELAY(Q, d) holds by showing that $Q(D \cup U, \tau_{in} - d) \subseteq Q(D, \tau_{in} - d)$ for every τ_{in}-history D, time point τ_{in}, and τ_{in}-update U. Let $\mathbf{o} \in Q(D \cup U, \tau_{in} - d)$ for any tuple $\mathbf{o}$, time point τ_{in}, τ_{in}-history D, and τ_{in}-update U. We assume without loss of generality that A does not occur in $D \cup U$ since A does not occur in Π_Q. We show $\mathbf{o} \in Q(D, \tau_{in} - d)$. Let $D' = \{A(\tau_{in} - d)\}$. Since $\Pi_Q \subseteq \Pi_1$, $\mathbf{o} \in Q(D \cup U, \tau_{in} - d)$ implies $\Pi_1 \cup D \cup U \models P_Q(\mathbf{o}, \tau_{in} - d)$; hence, $\mathbf{o} \in Q_1(D \cup D' \cup U, \tau_{in} - d)$ by rule (7). Then, $\mathbf{o} \in Q_2(D \cup D' \cup U, \tau_{in} - d)$ by our assumption. Note that $\Pi_2 \cup D \cup D' \cup U \not\models B(\tau)$ for any $\tau > \tau_{in}$ because B is IDB and can only be derived by rule (8), and A does not occur in $D \cup U$; hence no fact $P(\mathbf{c}, \tau)$ occurs in a derivation of $P_Q(\mathbf{o}, \tau_{in} - d)$ from $\Pi_2 \cup D \cup D' \cup U$, since predicate P occurs only in rule (9), which requires $B(\tau)$; hence $\mathbf{o} \in Q_2(D \cup D', \tau_{in} - d)$ because D is a τ_{in}-history and U is a τ_{in}-update; and hence $\mathbf{o} \in Q(D, \tau_{in} - d)$ by the construction of Q_2.

Assume $Q_1 \not\sqsubseteq Q_2$, and hence there is a tuple $\mathbf{o}$, a time point τ, and a dataset D such that $\mathbf{o} \in Q_1(D, \tau)$ and $\mathbf{o} \notin Q_2(D, \tau)$. Let D' contain each fact in D with time argument at most $\tau_{in} = \tau + d$, and let $U = D \setminus D'$—note that U is a τ_{in}-update. We show that DELAY(Q, d) does not hold by showing $\mathbf{o} \in Q(D' \cup U, \tau_{in} - d)$ and $\mathbf{o} \notin Q(D', \tau_{in} - d)$ (where $\tau_{in} - d = \tau$).

We first show $\mathbf{o} \in Q(D' \cup U, \tau_{in} - d)$. We have that $\mathbf{o} \in Q_1(D' \cup U, \tau_{in} - d)$ because $D = D' \cup U$; hence $\Pi_1 \cup D' \cup U \models P_Q(\mathbf{o}, \tau_{in} - d)$ by rule (7); hence $\mathbf{o} \in Q(D' \cup U, \tau_{in} - d)$ because Π_1 is Π_Q extended with rule (7), which derives G that does not occur in Π_Q.

We next show $\mathbf{o} \notin Q(D', \tau_{in} - d)$. Let D'' be the set consisting of each fact in D' with time argument τ satisfying $\tau_{in} - d - \mathrm{rad}(Q) \leq \tau$. Note that each time point in D'' has time argument τ satisfying $\tau_{in} - d - \mathrm{rad}(Q) \leq \tau \leq \tau_{in}$, since the time points in D' are at most τ_{in}. We have that $\mathbf{o} \notin Q_2(D, \tau_{in} - d)$ implies $\mathbf{o} \notin Q_2(D'', \tau_{in} - d)$ by monotonicity of entailment because $D'' \subseteq D$. We show by contraposition that $\mathbf{o} \notin Q_2(D'', \tau_{in} - d)$ implies $\mathbf{o} \notin Q(D'', \tau_{in} - d)$. Let δ be a derivation of $P_Q(\mathbf{o}, \tau_{in} - d)$ from $\Pi_Q \cup D''$. Let δ' be the derivation obtained from δ by first adding a fresh root labelled with the proper instance of rule (7) and having the root of δ as a child; then replacing each EDB predicate P from Π_Q with the corresponding P'; then, for each node v and for each atom $P'(\mathbf{c}, \tau)$ in the body of the label of v, we add a child to v labelled with the proper instance of rule (9); and finally, for each node having B in the body of its label, we add the proper instance of rule (8). We have that δ' is a derivation of $G(\mathbf{o}, \tau_{in} - d)$ from $\Pi_2 \cup D''$ because: *(i)* $A(\tau_{in} - d)$ is in D'' since $\mathbf{o} \in Q_1(D' \cup U, \tau_{in} - d)$, and *(ii)* there is an instance of rule (8) deriving $B(\tau)$ from $A(\tau_{in} - d)$ for each $P(\mathbf{c}, \tau) \in D''$, since we have that τ satisfies $\tau_{in} - d - \mathrm{rad}(Q) \leq \tau \leq \tau_{in}$ as observed before. Finally, $\mathbf{o} \notin Q_2(D'', \tau_{in} - d)$ implies $\mathbf{o} \notin Q_2(D', \tau_{in} - d)$ by Lemma 8. $\square$

Window

Lemma 11. *There exists a* LOGSPACE*-computable many-one reduction ϕ from containment of datalog queries to* WINDOW *such that, for every instance $I = \langle Q_1, Q_2 \rangle$ of query containment, the query in $\phi(I)$ is nonrecursive if Q_1 and Q_2 are nonrecursive.*

Proof. Let $I = \langle Q_1, Q_2 \rangle$ be an instance of query containment with Q_1 and Q_2 datalog queries. Without loss of generality, $P_{Q_1} = P_{Q_2} = G$. For any rigid n-ary IDB predicate P and $i \in \{1, 2\}$, let P_i be a fresh rigid n-ary IDB predicate uniquely associated with P and i. For any rigid n-ary EDB (resp., IDB) predicate P, let P^{t} be a fresh temporal $(n + 1)$-ary EDB (IDB) predicate uniquely associated with P. Let t be a time variable. For $i \in \{1, 2\}$, let Π_i be Π_{Q_i} after replacing each rigid n-ary IDB predicate P with P_i; let Π'_i be Π_i after replacing each rigid atom $P(\mathbf{u})$ with the temporal atom $P^{\mathrm{t}}(\mathbf{u}, t)$. Let A be a fresh temporal unary EDB predicate. Let Q be the query such that P_Q is a fresh temporal IDB predicate of the same arity as G_1^{t} (or, equivalently, as G_2^{t}), and Π_Q is $\Pi'_1 \cup \Pi'_2$ extended with the following rules:

$$A(t - 1) \wedge G_1^{\mathrm{t}}(\mathbf{x}, t) \rightarrow P_Q(\mathbf{x}, t) \tag{18}$$

$$G_2^{\mathrm{t}}(\mathbf{x}, t) \rightarrow P_Q(\mathbf{x}, t) \tag{19}$$

Clearly, Q can be constructed in logarithmic space w.r.t. the size of Q_1 and Q_2.

We show that $Q_1 \sqsubseteq Q_2$ iff WINDOW holds for $\phi(I) = \langle Q, 0, 0 \rangle$.

Assume that $Q_1 \sqsubseteq Q_2$ holds. We show that WINDOW holds for $\langle Q, 0, 0 \rangle$, by showing that $Q(D \cup U, \tau_{out}) \subseteq Q(D[\tau_{in}] \cup U, \tau_{out})$ for every dataset D, time points τ_{in} and τ_{out} such that $\tau_{out} > \tau_{in}$, and τ_{in}-update U. Let $\mathbf{o}$ be a tuple, let D be a dataset, let τ_{in} and τ_{out} be time points such that $\tau_{out} > \tau_{in}$, and let U be a τ_{in}-update such that $\mathbf{o} \in Q(D \cup U, \tau_{out})$. Let δ be a derivation of $P_Q(\mathbf{o}, \tau_{out})$ from $\Pi_Q \cup D \cup U$. The root of δ is labelled with an instance of either rule (18) or rule (19) since P_Q does not occur in $\Pi'_1 \cup \Pi'_2$. We consider the two cases separately. If the label is an instance of rule (18), then $\Pi_Q \cup D \cup U \models G_1^{\mathrm{t}}(\mathbf{o}, \tau_{out})$; then $\Pi'_1 \cup D \cup U \models G_1^{\mathrm{t}}(\mathbf{o}, \tau_{out})$ by the construction of Π_Q; then $\Pi'_1 \cup D[\tau_{in}] \cup U \models G_1^{\mathrm{t}}(\mathbf{o}, \tau_{out})$ because atoms in Π'_1 all have t as time argument and $\tau_{out} > \tau_{in}$ by assumption; then $\mathbf{o} \in Q_1(D')$ by the construction of Π'_1, where D' is the dataset consisting of each rigid fact $P(\mathbf{c})$ for $P^{\mathrm{t}}(\mathbf{c}, \tau_{out}) \in D[\tau_{in}] \cup U$; then $\mathbf{o} \in Q_2(D')$ since $Q_1 \sqsubseteq Q_2$ by assumption; then $\Pi'_2 \cup D[\tau_{in}] \cup U \models G_2^{\mathrm{t}}(\mathbf{o}, \tau_{out})$ by the constructions of Π'_2 and D'; then $\Pi_Q \cup D[\tau_{in}] \cup U \models G_2^{\mathrm{t}}(\mathbf{o}, \tau_{out})$ because $\Pi'_2 \subseteq \Pi_Q$; then $\mathbf{o} \in Q(D[\tau_{in}] \cup U, \tau_{out})$ by rule (19). If the root of δ is labelled with an instance of rule (19), then $\Pi_Q \cup D \cup U \models G_2^{\mathrm{t}}(\mathbf{o}, \tau_{out})$; then $\Pi'_2 \cup D \cup U \models G_2^{\mathrm{t}}(\mathbf{o}, \tau_{out})$ by the construction of Π_Q; then $\Pi'_2 \cup D[\tau_{in}] \cup U \models G_2^{\mathrm{t}}(\mathbf{o}, \tau_{out})$

because atoms in Π'_2 all have t as time argument and $\tau_{out} > \tau_{in}$ by assumption; then $\Pi_Q \cup D[\tau_{in}] \cup U \models G_2^{\mathrm{t}}(\mathbf{o}, \tau)$ because $\Pi'_2 \subseteq \Pi_Q$.

For the converse, assume $Q_1 \not\sqsubseteq Q_2$, and hence there is a tuple $\mathbf{o}$ of objects and a dataset D such that $\mathbf{o} \in Q_1(D)$ and $\mathbf{o} \notin Q_2(D)$. We show that WINDOW does not hold on $\langle Q, 0, 0 \rangle$. Let $D' = \{A(0)\}$, and let $\tau_{out} = 1$ and let $\tau_{in} = 0$. Let U be the τ_{in}-update containing each temporal fact $P^{\mathrm{t}}(\mathbf{c}, \tau_{out})$ for $P(\mathbf{c}) \in D$. We have that $\mathbf{o} \in Q_1(D)$ implies $\Pi_Q \cup U \models G_1^{\mathrm{t}}(\mathbf{o}, \tau_{out})$ by the construction of Π'_1. Hence $\mathbf{o} \in Q(D' \cup U, \tau_{out})$ by rule (18). We show that that $\mathbf{o} \notin Q(D'[\tau_{in}] \cup U, \tau_{out})$. Note that $D'[\tau_{in}] = \emptyset$. Let us assume by contradiction that $\mathbf{o} \in Q(U, \tau_{out})$. Let δ be a derivation of $P_Q(\mathbf{o}, \tau_{out})$ from $\Pi_Q \cup U$. The root of δ is labelled with an instance of either rule (18) or rule (19) since P_Q does not occur in $\Pi'_1 \cup \Pi'_2$. We discuss the two cases separately. If the root of δ is labelled with an instance of rule (18), then $A(\tau_{out} - 1) \in U$, which cannot be because U contains no time point different from τ_{out}. If the root of δ is labelled with an instance of rule (19), then $\Pi_Q \cup U \models G_2^{\mathrm{t}}(\mathbf{o}, \tau_{out})$; then $\Pi'_2 \cup U \models G_2^{\mathrm{t}}(\mathbf{o}, \tau_{out})$ by construction of Π_Q; then $\mathbf{o} \in Q_2(D)$ by construction of Π_2 and U, which contradicts our assumption. $\square$

Theorem 6. WINDOW *is undecidable.*

Proof. The claim follows by Lemma 11 since query containment for datalog is undecidable by the results in (Shmueli 1993). $\square$

Theorem 8. WINDOW *restricted to nonrecursive queries that are connected and constant-free is* co-NEXP-complete.

Proof. We first prove hardness. The reduction ϕ in Lemma 11 is such that, for each instance $I = \langle Q_1, Q_2 \rangle$ of query containment, the query in $\phi(I)$ is connected, and is also nonrecursive constant-free if Q_1 and Q_2 are nonrecursive constant-free. Furthermore, query containment is co-NEXP-hard already for nonrecursive constant-free datalog queries, by the results in (Benedikt and Gottlob 2010).

For the upper bound, we next show that there is a LOGSPACE-computable many-one reduction ϕ from WINDOW restricted to nonrecursive queries to nonrecursive query containment; the claim then follows by Lemma 5. Consider the construction given in Section 4.3. We argue that $Q_1 \sqsubseteq Q_2$ iff WINDOW(Q, d, s) holds.

For the direction from left to right, suppose $Q_1 \sqsubseteq Q_2$, and let D be a dataset, τ_{in}, τ_{out} time points such that $\tau_{out} > \tau_{in} - d$, and U a τ_{in}-update. We need to show $Q(D \cup U, \tau_{out}) = Q(D[\tau_{in} - s] \cup U, \tau_{out})$. Without loss of generality, we show the claim for $\tau_{in} = 0$ and $U = \emptyset$; since Q contains no time points, for each D', U' and τ'_{in} we have $Q(D, \tau_{out}) = Q(D' \cup U', \tau'_{out})$, where $\tau_{out} = \tau'_{out} - \tau'_{in}$ and D is obtained from $D' \cup U'$ by replacing each temporal fact $P(\mathbf{o}, \tau)$ with $P(\mathbf{o}, \tau - \tau'_{in})$; thus, $Q(D' \cup U', \tau'_{out}) = Q(D'[\tau'_{in} - s] \cup U', \tau'_{out})$ holds if and only if so does $Q(D, \tau_{out}) = Q(D[-s], \tau_{out})$.

The inclusion $Q(D[-s], \tau_{out}) \subseteq Q(D, \tau_{out})$ is immediate by monotonicity of entailment. For the other inclusion, suppose $\mathbf{o} \in Q(D, \tau_{out})$. We distinguish two cases.

If $\tau_{out} > -s + \mathrm{rad}(Q)$, the derivation of $P_Q(\mathbf{o}, \tau_{out})$ from $D \cup \Pi_Q$ does not involve facts at time points before $-s + 1$; this implies $\Pi_Q \cup D[-s] \models P_Q(\mathbf{o}, \tau_{out})$ and hence $\mathbf{o} \in Q(D[-s], \tau_{out})$.

Similarly, if $-d < \tau_{out} \leq -s + \mathrm{rad}(Q)$, we have $\Pi_Q \cup D' \models P_Q(\mathbf{o}, \tau_{out})$, where D' is obtained from $D[-d - \mathrm{rad}(Q)]$ by additionally removing all temporal facts holding after $-s + 2 \cdot \mathrm{rad}(Q)$, since no derivation of $P_Q(\mathbf{o}, \tau_{out})$ from $\Pi_Q \cup D$ involves facts at or before $-d - \mathrm{rad}(Q)$, or after $-s + 2 \cdot \mathrm{rad}(Q)$. Consider $D'' = D \cup \{In(0)\}$. By construction, we then have $\Pi^{d + \mathrm{rad}(Q)} \cup D'' \models P'(\mathbf{c}, \tau)$ if and only if $P(\mathbf{c}, \tau) \in D'$. Consequently, we have $\Pi^{d + \mathrm{rad}(Q)} \cup D'' \models P_Q(\mathbf{o}, \tau_{out})$, and, since $-d < \tau_{out} \leq -s + \mathrm{rad}(Q)$, also $\Pi^{d + \mathrm{rad}(Q)} \cup D'' \models Out(\tau_{out})$; thus, $\mathbf{o} \in Q_1(D'', \tau_{out})$. By assumption, we obtain $\mathbf{o} \in Q_2(D'', \tau_{out})$, i.e., $\Pi^s \cup D'' \models G(\mathbf{o}, \tau_{out})$, and hence $\Pi^s \cup D'' \models P_Q(\mathbf{o}, \tau_{out})$. By construction, any derivation of $P_Q(\mathbf{o}, \tau_{out})$ from $\Pi^s \cup D''$ can only involve temporal facts holding after $-s$. Thus, given a derivation δ of $P_Q(\mathbf{o}, \tau_{out})$ from $\Pi^s \cup D''$, by replacing each subtree of δ whose root is labelled by $P(\mathbf{c}', \tau') \wedge B(\tau') \rightarrow P'(\mathbf{c}', \tau')$ with the leaf $P(\mathbf{c}', \tau')$, we obtain a derivation of $P_Q(\mathbf{o}, \tau_{out})$ from $\Pi_Q \cup D[-s]$. Consequently, $\mathbf{o} \in Q(D[-s], \tau_{out})$.

The direction from right to left is similar. $\square$