Can you answer while you wait?

Luís Cruz-Filipe[1][0000−0002−7866−7484], Graça Gaspar[2][0000−0002−3106−0423], and Isabel Nunes[2][0000−0003−3966−4966]

[1] Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark lcfilipe@gmail.com
[2] LASIGE, Department of Informatics, Faculty of Sciences, University of Lisbon, Lisbon, Portugal {mdgaspar,minunes}@fc.ul.pt

Abstract. Continuous query answering is a challenging problem faced by systems that need to reason over data as it arrives. Recently, a logic-based approach to this problem has been proposed that advocates generating hypothetical query answers – potential answers that are consistent with the available data, but still require confirmation by future input. The current work studies hypothetical query answering in realistic settings, where data may arrive out of order. This requires revising its semantics and reanalysing the intuitions that led to the design of the existing algorithms, in order to develop a novel incremental online algorithm that takes into account that past data may yet arrive. We also discuss how our methods may be extended to channels with losses.

1 Introduction

Reasoning systems used in today's world must react in real time to information that they receive from e.g. sensors, continuously producing results in an online fashion. Conceptually, one of the ways to model this influx of information is as a data stream, and the reasoning tasks that these systems have to perform are usually called *continuous queries*.

One important line of work [10, 19, 42, 45, 49] views continuous queries as logic programs whose facts arrive through a data stream, and applies logic-based methods to compute answers to those queries. In practice, answers may take a long time to compute, and it may be useful to know that some answers are likely to be produced in the future, based on available information that might lead to their generation. This motivated the introduction of *hypothetical answers* [17]: answers supported by the information provided so far by the input stream, but that depend on other facts in the future being true.

Depending on the underlying communication system, information may be delayed, and may arrive unordered. The reasoning system must therefore allow for the possibility that absent data may still arrive, and if necessary delay the production of answers [7, 20, 46]. The approaches mentioned above abstract from communication delays by assuming that the data from the data stream is ordered: information with timestamp greater than t is "put on hold" until all data about timestamp t is produced. This assumption simplifies the theoretical development greatly, but is not easily implementable.

As soon as we remove the assumption that the data stream is ordered, the typical strategies for continuous query answering start failing, since they rely on the possibility of processing information grouped by time point. In the present work, we propose an alternative approach to computing and updating hypothetical answers that treats communication delays explicitly. We define a procedure for incrementally computing hypothetical answers in the presence of delays, by means of flexible strategies that deal with information as it arrives while acknowledging the possibility that older data may arrive later on. The only requirement is that a limit is known to how delayed the information may be, which we argue is reasonable in many practical applications. In the conclusions, we discuss how to remove this requirement.

Structure. Section 2 revisits the syntax of Temporal Datalog [45] and the main ideas behind the formalism of hypothetical answers [17], and introduces the running example that we use throughout this article. Section 3 updates the declarative and operational semantics of hypothetical answers to allow for communication delays. The new online algorithm for maintaining and updating hypothetical answers, given in Section 4, relies on the novel concept of *local mgu*. This section also includes proofs of soundness and completeness of the algorithm. Section 5 discusses alternative approaches to communication delays, and Section 6 includes some conclusions and directions for future work.

2 Background

2.1 Continuous queries in Temporal Datalog

We work in *Temporal Datalog*, the fragment of negation-free Datalog extended with the special temporal sort from [15]. The formalism for writing continuous queries over datastreams follows [45] with slight adaptations.

Syntax of Temporal Datalog. Temporal Datalog is an extension of Datalog [14] where constants and variables can have two sorts: *object* or *temporal*. Terms are also sorted: an *object term* is either an object constant or an object variable, and a *time term* is either a natural number, a time variable, or an expression of the form $T + \mathsf{k}$ where T is a time variable and k is an integer. Time constants are also called *timestamps*.

Predicates take exactly one temporal parameter, which is the last one. (Some works also allow predicates with no temporal parameter, but this adds no expressive power to the language.) Atoms, rules, facts and programs are defined as usual. Rules are assumed to be safe: each variable in the head must occur in the body.

A term, atom, rule, or program is *ground* if it contains no variables. We write $\mathsf{var}(\alpha)$ for the set of variables occurring in an atom α, and extend this function homomorphically to rules and sets. A *fact* is a function-free ground atom.

A predicate symbol is said to be *intensional* or IDB if it occurs in an atom in the head of a rule with non-empty body, and *extensional* or EDB if it is defined only through facts. This classification extends to atoms in the natural way.

Substitutions are functions mapping a finite set of variables to terms of the expected sort. Given a rule r and a substitution θ, the corresponding *instance* $r' = r\theta$ of r is obtained by simultaneously replacing every variable X in r by $\theta(X)$ and computing any additions of temporal constants.

A *temporal query* is a pair $Q = \langle P, \Pi \rangle$ where Π is a program and P is an IDB atom in the language underlying Π. We do not require P to be ground, and typically the temporal parameter is uninstantiated.[3]

A *dataset* is a family $D = \{D|_\tau \mid \tau \in \mathbb{N}\}$, where $D|_\tau$ represents the set of EDB facts delivered by a data stream at time point τ. Note that every fact in $D|_\tau$ has timestamp at most τ; facts with timestamp lower than τ correspond to communication delays. We call $D|_\tau$ the *τ-slice* of D, and define also the *τ-history* $D_\tau = \bigcup\{D|_{\tau'} \mid \tau' \leq \tau\}$. It follows that $D|_\tau = D_\tau \setminus D_{\tau-1}$ for every τ, and that D_τ also contains only facts whose temporal argument is at most τ. By convention, $D_{-1} = \emptyset$.

Semantics. The semantics of Temporal Datalog is defined in the usual way over Herbrand models, evaluating time terms to a natural number in the obvious way.

An *answer* to a query $Q = \langle P, \Pi \rangle$ over a set of ground facts S is a ground substitution θ over the set of variables in P such that $\Pi \cup S \models P\theta$. In this work, S is typically a τ-history of some dataset D. We denote the set of all answers to Q over D_τ as $\mathcal{A}(Q, D, \tau)$.

We illustrate these concepts with our running example, which is a variant of Example 1 in [45].

Example 1. The following program Π_E tracks activation of cooling measures in a set of wind turbines equipped with sensors, recording malfunctions and shutdowns, based on temperature readings $\mathsf{Temp}(\mathit{Device}, \mathit{Level}, \mathit{Time})$.

$$\begin{aligned}
\mathsf{Temp}(X, \mathsf{high}, T) &\rightarrow \mathsf{Flag}(X, T) \\
\mathsf{Flag}(X, T) \wedge \mathsf{Flag}(X, T+1) &\rightarrow \mathsf{Cool}(X, T+1) \\
\mathsf{Cool}(X, T) \wedge \mathsf{Flag}(X, T+1) &\rightarrow \mathsf{Shdn}(X, T+1) \\
\mathsf{Shdn}(X, T) &\rightarrow \mathsf{Malf}(X, T-2)
\end{aligned}$$

Malfunctions correspond to answers to the query $Q_E = \langle \mathsf{Malf}(X, T), \Pi_E \rangle$. If we assume $D_0 = \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0)\}$, then at time point 0 there is no answer to Q_E. If $\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1)$ arrives to D at time point 1, then $D_1 = D_0 \cup \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1)\}$, and there still is no answer to Q_E. Finally, the arrival of $\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 2)$ to D at time point 2 yields $D_2 = D_1 \cup \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 2)\}$, allowing us to infer $\mathsf{Malf}(\mathsf{wt2}, 0)$. Then $\{X := \mathsf{wt2}, T := 0\} \in \mathcal{A}(Q_E, D, 2)$. ◁

[3] The most common exception is if P represents a property that does not depend on time, where by convention the temporal parameter is instantiated to 0.

In this example, all facts were delivered instantly by the data stream. In the presence of communication delays, the answer $\{X := \mathsf{wt2}, T := 0\}$ might not be produced at time point 2, but only later.

2.2 SLD-resolution

Our development is based on SLD-resolution. We summarize the key relevant concepts and results, following [11, 37].

A *definite clause* is a disjunction of literals containing at most one positive literal. A definite clause with only negative literals is called a *goal* and written $\neg \wedge_j \beta_j$, where the β_j are atoms. Definite clauses with one positive literal are written in the standard rule notation $\wedge_i \alpha_i \rightarrow \alpha$, where the α_i and α are atoms.

A *most general unifier (mgu)* of two atomic formulas $P(\vec{X})$ and $P(\vec{Y})$ is a substitution θ such that: (i) $P(\vec{X})\theta = P(\vec{Y})\theta$ and (ii) for all σ, if $P(\vec{X})\sigma = P(\vec{Y})\sigma$ then $\sigma = \theta\gamma$ for some substitution γ. If C is a rule $\wedge_i \alpha_i \rightarrow \alpha$, G is a goal $\neg \wedge_j \beta_j$ with $\mathsf{var}(G) \cap \mathsf{var}(C) = \emptyset$, and θ is an mgu of α and β_k, then the *resolvent* of G and C using θ is the goal $\neg \left(\bigwedge_{j<k} \beta_j \wedge \bigwedge_i \alpha_i \wedge \bigwedge_{j>k} \beta_j \right) \theta$.

Let P be a program and G be a goal. An *SLD-derivation* of $P \cup \{G\}$ is a sequence $G_0, G_1, \ldots$ of goals, a sequence $C_1, C_2, \ldots$ of α-renamings of program clauses of P and a sequence $\theta_1, \theta_2, \ldots$ of substitutions such that $G_0 = G$ and G_{i+1} is the resolvent of G_i and C_{i+1} using θ_{i+1}. An *SLD-refutation* of $P \cup \{G\}$ is a finite SLD-derivation of $P \cup \{G\}$ ending in the empty clause ($\square$), and its *computed answer* is obtained by restricting the composition of $\theta_1, \ldots, \theta_n$ to the variables occurring in G.

SLD-resolution is sound and complete. If θ is a computed answer for $P \cup \{G\}$, then $P \models \neg(\forall G\theta)$. Conversely, if $P \models \neg(\forall G\theta)$, then there exist σ and γ such that $\theta = \sigma\gamma$ and σ is a computed answer for $P \cup \{G\}$. The *independence of the computation rule* states that the order in which the literals in the goal are resolved with rules from the program does not affect the existence of an SLD-refutation.

2.3 Hypothetical answers

In Example 1, the answer to the query Q_E could only be determined when $\tau = 2$. However, we could already infer that this answer might arise from the fact that $D_0 = \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0)\}$. The later delivery of $\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1)$ supports this possibility, while its absence from the data stream would have discarded it.

The formalism of *hypothetical answers* [17] builds on this idea. A hypothetical answer to a query $Q = \langle P, \Pi \rangle$ given a dataset D and a time instant τ is a pair $\langle \theta, H \rangle$ where θ is a substitution over the variables in P and H is a finite set of ground EDB atoms (the hypotheses) such that all atoms in H have a timestamp larger than τ, $\Pi \cup D_\tau \cup H \models P\theta$, and H is minimal with respect to set inclusion. Furthermore, if the minimal subset E of D_τ such that $\Pi \cup E \cup H \models P\theta$ is non-empty, then $\langle \theta, H \rangle$ is said to be *supported* by E (the *evidence*).

Example 2. In Example 1, $\langle [X := \mathsf{wt2}, T := 0], \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 2)\}\rangle$ is a hypothetical answer for Q_E and D at time 1. This answer is supported by the evidence $\{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0), \mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1)\}$.

The pair $\langle [X := \mathsf{wt2}, T := 3], \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, t) \mid t = 3, 4, 5\}\rangle$ is also a hypothetical answer for Q_E and D at time 1, but it is not supported. ◁

Hypothetical answers can be computed dynamically as information is delivered by the data stream. Given a query $Q = \langle P, \Pi\rangle$, an offline pre-processing step applies SLD-resolution to Π and $\leftarrow P$ until only EDB atoms remain. All leaves and corresponding substitutions are collected in a set $\mathcal{P}_Q$ of pairs $\langle \theta, H\rangle$.

For each time point τ, a set of schematic hypothetical answers $\mathcal{S}_\tau$ of the form $\langle \theta, E, H\rangle$ is computed online as follows.

1. If $\langle \theta, H\rangle \in \mathcal{P}_Q$ and σ is such that $H\sigma \setminus D|_\tau$ only contains atoms with timestamp higher than τ, then $\langle \theta\sigma, H\sigma \cap D|_\tau, H\sigma \setminus D|_\tau\rangle$ is added to $\mathcal{S}_\tau$.
2. If $\langle \theta, E, H\rangle \in \mathcal{S}_{\tau-1}$ and σ is such that $H\sigma \setminus D|_\tau$ only contains atoms whose timestamp is higher than τ, then the triple $\langle \theta\sigma, E \cup (H\sigma \cap D|_\tau), H\sigma \setminus D|_\tau\rangle$ is added to $\mathcal{S}_\tau$.

The candidate substitutions σ are easily obtained by unifying the elements of each set H that have minimal temporal argument with the elements of $D|_\tau$.

Example 3. In the context of our running example, the pre-processing step yields the singleton set

$$\mathcal{P}_{Q_E} = \{\langle \emptyset, \{\mathsf{Temp}(X, \mathsf{high}, T), \mathsf{Temp}(X, \mathsf{high}, T+1), \mathsf{Temp}(X, \mathsf{high}, T+2)\}\rangle\}.$$

Taking D_0, D_1 and D_2 as in Example 1, the previous algorithm yields

$$\begin{aligned}
\mathcal{S}_0 &= \{\langle [X := \mathsf{wt2}, T := 0], \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0)\}, \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, i) \mid i = 1, 2\}\rangle\}\\
\mathcal{S}_1 &= \{\langle [X := \mathsf{wt2}, T := 0], \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, i) \mid i = 0, 1\}, \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 2)\}\rangle,\\
&\quad \langle [X := \mathsf{wt2}, T := 1], \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1)\}, \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, i) \mid i = 2, 3\}\rangle\}\\
\mathcal{S}_2 &= \{\langle [X := \mathsf{wt2}, T := 0], \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, i) \mid i = 0, 1, 2\}, \emptyset\rangle,\\
&\quad \{\langle [X := \mathsf{wt2}, T := 1], \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, i) \mid i = 1, 2\}, \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 3)\}\rangle,\\
&\quad \langle [X := \mathsf{wt2}, T := 2], \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 2)\}, \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, i) \mid i = 3, 4\}\rangle\}
\end{aligned}$$

This algorithm is sound and complete: the supported hypothetical answers at each time point τ are the ground instantiations of the schematic answers computed at the same time point.

3 Introducing communication delays

So far, facts are assumed to be received instantaneously at the data stream: any fact with timestamp τ must be in $D|_\tau$, if present. However, communications take time, so a fact with timestamp τ may arrive later.

In this section, we extend the formalism of hypothetical answers to deal with these communication delays. We assume that communications may be delayed (but not lost) and that there is a known upper bound on the delay. In practice, there are communication protocols that ensure this property (with high enough probability).

The bound on the delay may be different for different predicates, and for different instantiations of the same predicate; the only restriction is that it may not depend on the timestamp.[4] We model this as a function δ mapping each ground EDB atom in the language of Π to a natural number, with the restriction that: if a, a' differ only in their temporal argument, then $\delta(a) = \delta(a')$. We extend δ to non-ground atoms by defining $\delta(P(t_1, \ldots, t_n))$ as the maximum of all $\delta(P(t'_1, \ldots, t'_n))$ such that $P(t'_1 \ldots, t'_n)$ is a ground instance of $P(t_1, \ldots, t_n)$, and to predicate symbols by $\delta(P) = \delta(P(X_1, \ldots, X_n))$.

3.1 Declarative semantics

The first ingredient in our extension is the following notion.

Definition 1. *A ground atom $P(t_1, \ldots, t_n)$ is* future-possible for τ *if $\tau < t_n + \delta(P(t_1, \ldots, t_n))$.*

In other words, an atom is future-possible for τ if it still may be delivered by the data stream. Future-possible atoms generalize the notion of "atom with timestamp greater than τ" – in particular, any such atom is future-possible for τ.

Hypothetical answers can now be generalized to include future-possible atoms.

Definition 2. *A* hypothetical answer *to query Q over D_τ is a pair $\langle \theta, H \rangle$, where θ is a substitution and H is a finite set of ground EDB atoms (the hypotheses) such that:*

- $\mathsf{supp}(\theta) = \mathsf{var}(()P)$*, i.e., θ only changes variables that occur in P;*
- *H only contains atoms future-possible for τ;*
- *$\Pi \cup D_\tau \cup H \models P\theta$;*
- *H is minimal with respect to set inclusion.*

$\mathcal{H}(Q, D, \tau)$ is the set of hypothetical answers to Q over D_τ.

As before, if the minimal subset E of D_τ such that $\Pi \cup E \cup H \models P\theta$ is non-empty, then $\langle \theta, H, E \rangle$ is a supported answer to Q over D_τ. We denote the set of all supported answers to Q over D_τ by $\mathcal{E}(Q, D, \tau)$.

The key properties of hypothetical and supported answers still hold for this generalized notion. The proofs are straightforward adaptations of those in [16].

[4] This is a reasonable assumption e.g. in the case of information originating from sensors, where the delay may depend on the distance and infrastructure and therefore be different for each sensor, but typically does not change over time. At the end of this article, we briefly discuss how to extend our framework to deal with unbounded communication delays and possible loss of information.

Proposition 1. *Let $Q = \langle P, \Pi \rangle$ be a query, D be a dataset and τ be a time instant. If $\langle \theta, \emptyset \rangle \in \mathcal{H}(Q, D, \tau)$, then $\theta \in \mathcal{A}(Q, D, \tau)$.*

Proposition 2. *Let $Q = \langle P, \Pi \rangle$ be a query, D be a dataset and τ be a time instant. If $\langle \theta, H \rangle \in \mathcal{H}(Q, D, \tau)$, then there exist a time point $\tau' \geq \tau$ and a dataset D' such that $D_\tau = D'_\tau$ and $\theta \in \mathcal{A}(Q, D', \tau')$.*

Proposition 3. *Let $Q = \langle P, \Pi \rangle$ be a query, D be a dataset and τ be a time instant. If $\langle \theta, H \rangle \in \mathcal{H}(Q, D, \tau)$, then there exists H^0 such that $\langle \theta, H^0 \rangle \in \mathcal{H}(Q, D, -1)$ and $H = H^0 \setminus D_\tau$. Furthermore, if $H \neq H^0$, then $\langle \theta, H, H^0 \setminus H \rangle \in \mathcal{E}(Q, D, \tau)$.*

Proposition 4. *Let $Q = \langle P, \Pi \rangle$ be a query, D be a dataset and τ be a time instant. If $\langle \theta, H \rangle \in \mathcal{H}(Q, D, \tau)$ and $\tau' < \tau$, then there exists $\langle \theta, H' \rangle \in \mathcal{H}(Q, D, \tau')$ such that $H = H' \setminus (D_\tau \setminus D_{\tau'})$.*

Example 4. We illustrate these concepts with the program from Example 1. We assume that $\delta(\mathsf{Temp}) = 1$, that $D|_0 = \{\mathsf{Temp}(\mathsf{wt4}, \mathsf{high}, 0)\}$, and that $D|_1 = \emptyset$. Let $\theta = [X := \mathsf{wt4}, T := 0]$. Then

$$\langle \theta, \{\mathsf{Temp}(\mathsf{wt4}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt4}, \mathsf{high}, 2)\} \rangle \in \mathcal{H}(Q_E, D, 1)\,,$$

reflecting the intuition that $\mathsf{Temp}(\mathsf{wt4}, \mathsf{high}, 1)$ may still arrive in $D|_2$. This answer is supported by $\mathsf{Temp}(\mathsf{wt4}, \mathsf{high}, 0)$. ◁

3.2 Operational semantics

The operational semantics based on SLD-resolution, which forms the basis for the online algorithm in [17], also extends to a scenario with communication delays: the key change is definining an operational counterpart to the notion of future-possible atom.

Definition 3. *An atom $P(t_1, \ldots, t_n)$ is a* potentially future atom wrt τ *if either t_n contains a temporal variable or t_n is ground and $\tau < t_n + \delta(P(t_1, \ldots, t_n))$.*

This concept generalizes the notion of future-possible atom for τ to possibly non-ground atoms. In particular, any atom whose temporal parameter contains a variable is potentially future, since it may be instantiated to a future timestamp.

Without loss of generality, we assume that all SLD-derivations have the following property: when unifying a goal $\neg \wedge_i \alpha_i$ with a clause $\wedge_j \beta_j \to \beta$, the chosen mgu $\theta = [X_1 := t_1, \ldots, X_n := t_n]$ is such that all variables occurring in $t_1, \ldots, t_n$ also occur in the chosen atom α_k.

Definition 4. *An* SLD-refutation with future premises *of Q over D_τ is a finite SLD-derivation of $\Pi \cup D_\tau \cup \{\neg P\}$ whose last goal only contains potentially future EDB atoms wrt τ.*

If $\mathcal{R}$ is an SLD-refutation with future premises of Q over D_τ with last goal $G = \neg \wedge_i \alpha_i$ and θ is the substitution obtained by restricting the composition of the mgus in $\mathcal{R}$ to $\mathsf{var}(P)$, then $\langle \theta, \wedge_i \alpha_i \rangle$ is a computed answer with premises *to Q over D_τ, denoted $\langle Q, D_\tau \rangle \vdash_{\mathsf{SLD}} \langle \theta, \wedge_i \alpha_i \rangle$.*

Example 5. Consider the setting of Example 4 and $\tau = 1$. There is an SLD-derivation of $\Pi \cup D_1 \cup \{\neg \mathsf{Malf}(X,T)\}$ ending with

$$\leftarrow \mathsf{Temp}(\mathsf{wt4}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt4}, \mathsf{high}, 2)\,,$$

which contains two potentially future EDB atoms with respect to 1. Thus,

$$\langle Q_E, D_1 \rangle \vdash_{\mathsf{SLD}} \langle \theta, \mathsf{Temp}(\mathsf{wt4}, \mathsf{high}, 1) \wedge \mathsf{Temp}(\mathsf{wt4}, \mathsf{high}, 2) \rangle$$

with $\theta = [X := \mathsf{wt4}, T := 0]$. ◁

As before, computed answers with premises are the operational counterpart to hypothetical answers, with two caveats: a computed answer with premises need not be ground, as the corresponding SLD-derivation may include some universally quantified variables in the last goal; and $\wedge_i \alpha_i$ may contain redundant conjuncts, in the sense that they might not be needed to establish the goal (see the examples in [17]).

With the new definitions, computed answers with premises and hypothetical answers are related as before.

Proposition 5 (Soundness). *Let $Q = \langle P, \Pi \rangle$ be a query, D be a dataset and τ be a time instant. Assume that $\langle Q, D_\tau \rangle \vdash_{\mathsf{SLD}} \langle \theta, \wedge_i \alpha_i \rangle$. Let σ be a ground substitution such that: (i) $\mathsf{supp}(\sigma) = \mathsf{var}(\wedge_i \alpha_i) \cup (\mathsf{var}(P) \setminus \mathsf{supp}(\theta))$ and (ii) if α_i is $P(t_1, \ldots, t_n)$, then $t_n\sigma + \delta(P(t_1, \ldots, t_n)) > \tau$. Then there is a set $H \subseteq \{\alpha_i\sigma\}_i$ such that $\langle (\theta\sigma)|_{\mathsf{var}(P)}, H \rangle \in \mathcal{H}(Q, D, \tau)$.*

Proposition 6 (Completeness). *Let $Q = \langle P, \Pi \rangle$ be a query, D be a dataset and τ be a time instant. If $\langle \theta, H \rangle \in \mathcal{H}(Q, D, \tau)$, then there exist substitutions ρ and σ and a finite set of atoms $\{\alpha_i\}_i$ such that $\theta = \rho\sigma$, $H = \{\alpha_i\sigma\}_i$ and $\langle Q, D_\tau \rangle \vdash_{\mathsf{SLD}} \langle \rho, \wedge_i \alpha_i \rangle$.*

The incremental algorithm in [17] is based on the idea of "organizing" SLD-derivations adequately to pre-process Π independently of D_τ, so that the computation of (hypothetical) answers can be split into an offline part and a less expensive online part. The following result asserts that this can be done.

Proposition 7. *Let $Q = \langle P, \Pi \rangle$ be a query and D be a dataset. For any time constant τ, if $\langle Q, D_\tau \rangle \vdash_{\mathsf{SLD}} \langle \theta, \wedge_i \alpha_i \rangle$, then there exist an SLD-refutation with future premises of Q over D_τ computing $\langle \theta, \wedge_i \alpha_i \rangle$ and a sequence $k_{-1} \leq k_0 \leq \ldots \leq k_\tau$ such that:*

- *goals $G_1, \ldots, G_{k_{-1}}$ are obtained by resolving with clauses from Π;*
- *for $0 \leq i \leq \tau$, goals $G_{k_{i-1}+1}, \ldots, G_{k_i}$ are obtained by resolving with clauses from $D|_i$.*

Proof. Immediate consequence of the independence of the computation rule. □

An SLD-refutation with the structure described in Proposition 7, which we call *stratified*, also has the property that all goals after $G_{k_{-1}}$ are always resolved with EDB atoms. Furthermore, each goal G_{k_i} contains only potentially future EDB atoms with respect to i. Let θ_i be the restriction of the composition of all substitutions in the SLD-derivation up to step k_i to $\mathsf{var}(P)$. Then $G_{k_i} = \neg \wedge_j \alpha_j$ represents all hypothetical answers to Q over D_i of the form $\langle (\theta_i \sigma)|_{\mathsf{var}(P)}, \wedge_j \alpha_j \rangle$ for some ground substitution σ (cf. Proposition 5).

This yields an online procedure to compute supported answers to continuous queries over data streams. In a pre-processing step, we calculate all computed answers with premises to Q over D_{-1}, and keep the ones with minimal set of formulas. (Note that Proposition 6 guarantees that all minimal sets are generated by this procedure, although some non-minimal sets may also appear.) The online part of the procedure then resolves each of these sets with the facts delivered by the data stream, adding the resulting resolvents to a set of schemata of supported answers (i.e. where variables may still occur). By Proposition 7, if there is at least one resolution step at this stage, then the hypothetical answers represented by these schemata all have evidence, so they are indeed supported.

The pre-processing step coincides exactly with that described in [17], which also includes examples and sufficient conditions for termination. The online part, though, needs to be reworked in order to account for communication delays. This is the topic of the next section.

4 Incremental computation of hypothetical answers

Pre-processing a query returns a finite set $\mathcal{P}_Q$ that represents $\mathcal{H}(Q, D, -1)$: for each computed answer $\langle \theta, \wedge_i \alpha_i \rangle$ with premises to Q over D_{-1} where $\{\alpha_i\}_i$ is minimal, $\mathcal{P}_Q$ contains an entry $\langle \theta, \{\alpha_i\}_i \rangle$. Each tuple $\langle \theta, H \rangle \in \mathcal{P}_Q$ represents the set of all hypothetical answers $\langle \theta\sigma, H\sigma \rangle$ as in Proposition 5. We now show how to compute and update the set $\mathcal{E}(Q, D, \tau)$ schematically.

Definition 5. *Let Γ and Δ be sets of atoms such that all atoms in Δ are ground. A substitution σ is a* local mgu *for Γ and Δ if: for every substitution θ such that $\Gamma\theta \cap \Delta = \Gamma\sigma \cap \Delta$ there exists another substitution ρ such that $\theta = \sigma\rho$.*

Intuitively, a local mgu for Γ and Δ is a substitution that behaves as an mgu of some subsets of Γ and Δ. Local mgus allow us to postpone instantiating some atoms, in order to cope with delayed information that needs to be processed later – see Example 6 below.

Lemma 1. *If Γ and Δ are finite, then the set of local mgus for Γ and Δ is computable.*

Proof. We claim that the local mgus for Γ and Δ are precisely the computed substitutions at some node of an SLD-tree for $\neg \bigwedge \Gamma$ and Δ.

Suppose σ is a local mgu for Γ and Δ, and let $\Psi = \{a \in \Gamma | a\sigma \in \Delta\}$. Build an SLD-derivation by unifying at each stage an element of Ψ with an element of

Δ. This is trivially a valid SLD-derivation, and the substitution θ it computes must be σ: (i) by soundness of SLD-resolution, θ is an mgu of Ψ and a subset of Δ; but $\Psi\theta \cap \Delta = \Delta$ is a set of ground atoms, so θ must coincide with σ on all variables occurring in Ψ, and be the identity on all other variables; and (ii) by construction $\Gamma\theta \cap \Delta = \Gamma\sigma \cap \Delta$, so θ cannot instantiate fewer variables than σ.

Consider now an arbitrary SLD-derivation for $\neg \bigwedge \Gamma$ and Δ with computed substitution σ, and let Ψ be the set of elements of Γ that were unified in this derivation. Then σ is an mgu of Ψ and a subset of Δ, and as before this means that it maps every variable in Ψ to a ground term and every other variable to itself. Suppose that θ is such that $\Gamma\theta \cap \Delta = \Psi\theta \cap \Delta$. In particular, θ also unifies Ψ and a subset of Δ, so it must coincide with σ in all variables that occur in Ψ. Taking ρ as the restriction of θ to the variables outside Ψ shows that σ is a local mgu for Γ and Δ. □

Example 6. Consider the program Π consisting of the rule

$$q(X,T) \leftarrow p(X,T), r(Y,T)\,,$$

where p is an EDB with $\delta(p) = 2$ and r is an EDB with $\delta(r) = 0$, and the query $\langle q(X,T), \Pi\rangle$. The pre-processing step for this query yields the singleton set $\mathcal{P}_Q = \{\langle \emptyset, \{p(X,T), r(Y,T)\}\rangle\}$.

Suppose that $D|_0 = \{p(\mathsf{a},0), r(\mathsf{b},0)\}$. There are two SLD-trees for the goal $\leftarrow p(X,T), r(Y,T)$ and $D|_0$:

$$\begin{array}{ccc} \leftarrow p(X,T), r(Y,T) & & \leftarrow p(X,T), r(Y,T) \\ \downarrow{\scriptstyle [X:=\mathsf{a},T:=0]} & & \downarrow{\scriptstyle [Y:=\mathsf{b},T:=0]} \\ \leftarrow r(Y,0) & & \leftarrow p(X,0) \\ \downarrow{\scriptstyle [Y:=\mathsf{b}]} & & \downarrow{\scriptstyle [X:=\mathsf{a}]} \\ \square & & \square \end{array}$$

These trees include the computed substitutions $\sigma_0 = \emptyset$, $\sigma_1 = [X := \mathsf{a}, T := 0]$, $\sigma_2 = [Y := \mathsf{b}, T := 0]$ and $\sigma_3 = [X := \mathsf{a}, Y := \mathsf{b}, T := 0]$, which are easily seen to be local mgus.

Although σ_3 is an answer to the query, we also need to consider substitution σ_2: since $\delta(p) = 2$, it may be the case that additional answers are produced (e.g. if $p(\mathsf{c},0) \in D|_2$). However, since $\delta(r) = 0$, substitution σ_1 can be safely discarded. Substitution σ_0 does not need to be considered: since it does not unify any element of H with $D|_0$, any potential answers it might produce would be generated by step 1 of the algorithm in future time points. ◁

Definition 6. *The set $\mathcal{S}_\tau$ of* schematic supported answers *for query Q at time τ is defined as follows.*

- $\mathcal{S}_{-1} = \{\langle \theta, \emptyset, H\rangle \mid \langle \theta, H\rangle \in \mathcal{P}_Q\}$.
- *If $\langle \theta, E, H\rangle \in \mathcal{S}_{\tau-1}$ and σ is a local mgu for H and $D|_\tau$ such that $H\sigma \setminus D|_\tau$ only contains potentially future atoms wrt τ, then $\langle \theta\sigma, E \cup E', H\sigma \setminus D|_\tau\rangle \in \mathcal{S}_\tau$, where $E' = H\sigma \cap D|_\tau$.*

Example 7. We illustrate this mechanism in the setting of Example 4, where

$$\mathcal{P}_Q = \{\langle \emptyset, \underbrace{\{\mathsf{Temp}(X, \mathsf{high}, T), \mathsf{Temp}(X, \mathsf{high}, T+1), \mathsf{Temp}(X, \mathsf{high}, T+2)\}}_{H}\rangle\}$$

and we assume that $\delta(\mathsf{Temp}) = 1$. We start by setting $\mathcal{S}_{-1} = \{\langle \emptyset, \emptyset, H\rangle\}$.

Since $D|_0 = \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0)\}$, SLD-resolution between H and $D|_0$ yields the local mgus $\emptyset$ and $[X := \mathsf{wt2}, T := 0]$. Therefore,

$$\mathcal{S}_0 = \{\langle \emptyset, \emptyset, \{\mathsf{Temp}(X, \mathsf{high}, T), \mathsf{Temp}(X, \mathsf{high}, T+1), \mathsf{Temp}(X, \mathsf{high}, T+2)\}\rangle, \\ \langle [X := \mathsf{wt2}, T := 0], \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0)\}, \{\underbrace{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, i) \mid i = 1, 2}_{H_0}\}\rangle\}.$$

Next, $D|_1 = \emptyset$, so trivially $D|_1 \subseteq H_0$ and therefore the empty substitution is a local mgu of H_0 and $D|_1$. Furthermore, H_0 only contains potentially future atoms wrt 1 because $\delta(\mathsf{Temp}) = 1$. The same argument applies to $D|_1$ and H. So $\mathcal{S}_1 = \mathcal{S}_0$.

We now consider several possibilities for what happens to the schematic supported answer $\langle [X := \mathsf{wt2}, T := 0], \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0)\}, H_0\rangle$ at time instant 2. Since H_0 is ground, the only local mgu of H_0 and $D|_2$ will always be $\emptyset$.

- If $\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1) \notin D|_2$, then $H_0 \setminus D|_2$ contains $\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1)$, which is not a potentially future atom wrt 2, and therefore this schematic supported answer is discarded.
- If $\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1) \in D|_2$ but $\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 2) \notin D|_2$, then $H_0 \setminus D|_2 = \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 2)\}$, which only contains potentially future atoms wrt 2, and therefore $\mathcal{S}_2$ contains the schematic supported answer

$$\langle [X := \mathsf{wt2}, T := 0], \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, i) \mid i = 0, 1, 2\}\rangle.$$

- Finally, if $\{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 2)\} \subseteq D|_2$, then $H_2 \setminus D|_2 = \emptyset$, and the system can output the answer $[X := \mathsf{wt2}, T := 0]$ to the original query. In this case, this answer would be added to $\mathcal{S}_2$, and then trivially copied to all subsequent $\mathcal{S}_\tau$. ◁

As in this example, answers are always propagated from $\mathcal{S}_\tau$ to $\mathcal{S}_{\tau+1}$. In an actual implementation of this algorithm, we would likely expect these answers to be output when generated, and discarded afterwards.

Proposition 8 (Soundness). *If $\langle \theta, E, H\rangle \in \mathcal{S}_\tau$, $E \neq \emptyset$, and σ instantiates all free variables in $E \cup H$, then $\langle \theta\sigma, H', E\sigma\rangle \in \mathcal{E}(Q, D, \tau)$ for some $H' \subseteq H\sigma$.*

Proof. By induction on τ, we show that $\langle Q, D_\tau\rangle \vdash_{\mathsf{SLD}} \langle \theta, \wedge_i \alpha_i\rangle$ with $H = \{\alpha_i\}_i$. For $\mathcal{S}_{-1}$ this is trivially the case, due to the way that $\mathcal{P}_Q$ is computed.

Assume now that there exists $\langle \theta_0, E_0, H_0\rangle \in \mathcal{S}_{\tau-1}$ such that σ_0 is a local mgu of H_0 and $D|_\tau$, $\theta = \theta_0\sigma_0$, $E = E_0 \cup (H_0\sigma_0 \cap D|_\tau)$ and $H = H_0\sigma_0 \setminus D|_\tau$. If $H_0\sigma_0 \cap D_\tau = \emptyset$, then necessarily $\sigma_0 = \emptyset$ (no steps of SLD-resolution were performed), and the thesis holds by induction hypothesis. Otherwise, we can

build the required derivation by extending the derivation obtained by induction hypothesis with unification steps between the relevant elements of H_0 and $D|_\tau$. By Lemma 1, this derivation computes the substitution θ.

Applying Proposition 5 to this SLD-derivation yields an $H' \subseteq H\sigma$ such that $\langle \theta\sigma, H' \rangle \in \mathcal{H}(Q, D, \tau)$. By construction, $E\sigma \neq \emptyset$ is evidence for this answer. □

It may be the case that $\mathcal{S}_\tau$ contains some elements that do not correspond to hypothetical answers because of the minimality requirement. Consider a simple case of a query Q where

$$\mathcal{P}_Q = \{\langle \emptyset, \{p(a,0), p(b,1), p(c,2)\}\rangle, \langle \emptyset, \{p(a,0), p(c,2), p(d,3)\}\rangle\},$$

and suppose that $D|_0 = \{p(a,0)\}$ and $D|_1 = \{p(b,1)\}$. Then

$$\mathcal{S}_1 = \{\langle \emptyset, \{p(a,0), p(b,1)\}, \{p(c,2)\}\rangle, \langle \emptyset, \{p(a,0)\}, \{p(c,2), p(d,3)\}\rangle\},$$

and the second element of this set has a non-minimal set of hypotheses. For simplicity, we do not to include a test for set inclusion in the definition of $\mathcal{S}_\tau$.

Proposition 9 (Completeness). *If $\langle \sigma, H, E \rangle \in \mathcal{E}(Q, D, \tau)$, then there exist a substitution ρ and a triple $\langle \theta, E', H' \rangle \in \mathcal{S}_\tau$ such that $\sigma = \theta\rho$, $H = H'\rho$ and $E = E'\rho$.*

Proof. By Proposition 6, $\langle Q, D_\tau \rangle \vdash_{\mathsf{SLD}} \langle \theta, \wedge_i \alpha_i \rangle$ for some substitution ρ and set of atoms $H' = \{\alpha_i\}_i$ with $H = \{\alpha_i\rho\}_i$ and $\sigma = \theta\rho$ for some θ. By Proposition 7, there is a stratified SLD-derivation computing this answer. The sets of atoms from $D|_\tau$ unified in each stratum of this derivation define the set of elements from H that need to be unified to construct the corresponding element of $\mathcal{S}_\tau$. □

The use of local mgus gives a worst-case exponential complexity to the computation of $\mathcal{S}_\tau$ from $\mathcal{S}_{\tau-1}$: since every node of the SLD-tree can give a local mgu as in the proof of Lemma 1, there can be as many local mgus as subsets of H that can be unified with $D|_\tau$. In practice, the number of substitutions that effectively needs to be considered is at most $(k+1)^v$, where v is the number of variables in H and k is the maximum number of possible instantiations for a variable in $D|_\tau$. In practice, v only depends on the program Π, and is likely to be small, while k is 0 if the elements of H are already instantiated.

If there are no communication delays and the program satisfies an additional property, we can regain the polynomial complexity from [17].

Definition 7. *A query $Q = \langle P, \Pi \rangle$ is* connected *if each rule in P contains at most one temporal variable, which occurs in the head if it occurs in the body.*

Proposition 10. *If the query Q is connected and $\delta(P) = 0$ for every predicate symbol P, then $\mathcal{S}_\tau$ can be computed from $\mathcal{P}_Q$ and $\mathcal{S}_{\tau-1}$ in time polynomial in the size of $\mathcal{P}_Q$, $\mathcal{S}_{\tau-1}$ and $D|_\tau$.*

Proof. Suppose that $\delta(P) = 0$ for every P.

If $\langle \theta, \emptyset, H \rangle \in \mathcal{S}_{\tau-1}$, then (i) $\emptyset$ is always a local mgu of H and $D|_\tau$, so this schematic answer can be added to $\mathcal{S}_\tau$ (in constant time); and (ii) we can compute in polynomial time the set $M \subseteq H$ of elements that need to be in $D|_\tau$ in order to yield a non-empty local mgu σ of H and $D|_\tau$ such that $H\sigma \setminus D|_\tau$ only contains potentially future atoms wrt τ – these are the elements whose timestamps are less or equal to all other elements' timestamps. (Since there are no delays, they must all arrive simultaneously at the data stream.) To decide which substitutions make M a subset of $D|_\tau$, we can perform classical SLD-resolution between M and $D|_\tau$. For each such element of $\mathcal{S}_{\tau-1}$, the size of every SLD-derivation that needs to be constructed is bound by the number of atoms in the initial goal, since $D|_\tau$ only contains facts. Furthermore, all unifiers can be constructed in time linear in the size of the formulas involved, since the only function symbol available is addition of temporal terms. Finally, the total number of SLD-derivations that needs to be considered is bound by the size of $\mathcal{S}_{\tau-1} \times D|_\tau$.

If $\langle \theta, E, H \rangle \in \mathcal{S}_{\tau-1}$ and $E \neq \emptyset$, then we observe that the temporal argument is instantiated in all elements of E and H – this follows from connectedness and the fact that the elements of E are ground by construction. Therefore, we know exactly which facts in H must unify with $D|_\tau$ – these are the elements of H whose timestamp is exactly τ. As above, the elements that must be added to $\mathcal{S}_\tau$ can then be computed in polynomial time by SLD-resolution. □

The key step of this proof amounts to showing that the algorithm from [17] computes all local mgus that can lead to schematic supported answers.

Both conditions stated above are necessary. If Q is not connected, then $\mathcal{S}_\tau$ may contain elements $\langle \theta, E, H \rangle$ where $E \neq \emptyset$ and H contains elements with uninstantiated timestamps; and if for some predicate $\delta(P) \neq 0$, then we must account for the possibility that facts about P arrive with some delay to the data stream. In either case, this means that we do not know the set of elements that need to unify with the data stream, and need to consider all possibilities.

Example 8. In the context of our running example, we say that a turbine has a manufacturing defect if it exhibits two specific failures during its lifetime: at some time it overheats, and at some (different) time it does not send a temperature reading. Since this is a manufacturing defect, we set it to hold at timepoint 0, regardless of when the failures occur. We model this property by the rule

$$\mathsf{Temp}(X, \mathsf{high}, T_1), \mathsf{Temp}(X, \mathsf{n/a}, T_2) \rightarrow \mathsf{Defective}(X, 0)\,.$$

Let Π'_E be the program obtained from Π_E by adding this rule, and consider now the query $Q' = \langle \mathsf{Defective}(X, T), \Pi'_E \rangle$. Performing SLD-resolution between Π'_E and $\mathsf{Defective}(X, 0)$ yields the goal $\neg\,(\mathsf{Temp}(X, \mathsf{high}, T_1) \wedge \mathsf{Temp}(X, \mathsf{n/a}, T_2))$, which only contains potentially future atoms with respect to -1.

Assume that $\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0) \in D_0$. Then

$$\langle \theta' = [X := \mathsf{wt2}, T := 0]\,, \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0)\}, \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{n/a}, T_2)\} \rangle \in \mathcal{S}_0\,.$$

We do not know if or when θ' will become an answer to the original query, but our algorithm is still able to output relevant information to the user. ◁

Under some assumptions, we can also bound the interval in which a schematic hypothetical answer can be present in $\mathcal{S}$. Assume that the query Q has a temporal variable T and that $\mathcal{S}_\tau$ contains a triple $\langle \theta, E, H \rangle$ with $T\theta \leq \tau$ and $H \neq \emptyset$.

- Suppose that there is a number d such that for every substitution σ and every $\tau \geq T\sigma + d$, $\sigma \in \mathcal{A}(Q, D, \tau)$ iff $\sigma \in \mathcal{A}(Q, D, T\sigma + d)$. Then the timestamp of each element of H is at most $\tau + d$.
- Similarly, suppose that there is a natural number w such that each σ is an answer to Q over D_τ iff σ is an answer to Q over $D_\tau \setminus D_{\tau - w}$. Then all elements in E must have timestamp at least $\tau - w$.

The values d and w are known in the literature as *(query) delay* and *window size*, respectively, see e.g. [45].

5 Related work

This work contributes to the field of stream reasoning, the task of conjunctively reasoning over streaming data and background knowledge [47].

Research advances on Complex Event Processors and Data Stream Management Systems [18], together with Knowledge Representation and the Semantic Web, all contributed to the several stream reasoning languages, systems and mechanisms proposed during the last decade [20].

Computing answers to a query over a data source that is continuously producing information, be it at slow or very fast rates, asks for techniques that allow for some kind of *incremental evaluation*, in order to avoid reevaluating the query from scratch each time a new tuple of information arrives. Several efforts have been made in that direction, capitalising on incremental algorithms based on seminaive evaluation [2, 8, 26, 27, 40], on truth maintenance systems [9], window oriented [23] among others. The framework we build upon [17] fits naturally in the first class, as it is an incremental variant of SLD-resolution.

Hypothetical query answering over streams is broadly related to works that study abduction in logic programming [21, 28], namely those that view negation in logic programs as hypotheses and relate it to contradiction avoidance [4, 22]. Furthermore, our framework can be characterized as applying an incremental form of data-driven abductive inference. Such forms of abduction have been used in other works [39], but with a rather different approach and in the plan interpretation context. To our knowledge, hypothetical or abductive reasoning has not been previously used in the context of stream reasoning, to answer continuous queries, although it has been applied to annotation of stream data [5].

A similar problem also arises in the area of incomplete databases, where the notions of possible and certain answers have been developed [30]. In this context, possible answers are answers to a complete database D' that the incomplete database D can represent, while certain answers consist of the tuples that belong to all complete databases that D can represent (the intersection of all possible answers). Libkin [36] the way those notions were defined, exploring

an an alternative way of looking at incomplete databases that dates back to Reiter [44], and that views a database as a logical theory. Libkin's approach was to explore the semantics of incompleteness further in a setting that is independent of a particular data model, and appealing to orderings to describe the degree of incompleteness of a data model and relying on those orderings to define certain answers. Other authors [24, 32, 33, 43] have also investigated ways to assign confidence levels to the information output to the user.

Most theoretical approaches to stream-processing systems commonly require input streams to be ordered. However, some approaches, namely from the area of databases and event processing, have developed techniques to deal with out-of-order data. An example of such a technique requires inserting special marks in the input stream (punctuations) to guide window processing [34]. Punctuations assert a timestamp that is a lower bound for all future incoming values of a attribute. Another technique starts by the potential generation of out-of-order results, which are then ordered by using stackbased data structures and associated purge algorithms [35].

Commercial systems [29] for continuous query answering use other techniques to deal with communication delays, not always with a solid foundational background. One such technique is the use of watermarks [3], which specify how long the system should wait for information to arrive. In practice, watermarks serve the same purpose as the function δ in our framework. However, they do not come with guarantees of correctness: in fact, information may arrive after the time point specified by the watermark, in which case it is typically discarded.

Memory consumption is a concern in [48], where a sound and complete stream reasoning algorithm is presented for a fragment of datalogMTL – forward-propagating datalogMTL – that also disallows propagation of derived information towards the past. DatalogMTL is a Datalog extension of the Horn fragment of MTL [31, 6], which was proposed in [13] for ontology-based access to temporal log data. DatalogMTL rules allow propagation of derived information to both past and future time points. Concerned with the possibly huge or unbounded set of input facts that have to be kept in memory, the authors of [48] restrict the set of operators of DatalogMTL to one that generates only so-called forward-propagating rules. They present a sound and complete algorithm for stream reasoning with forward-propagating datalogMTL that limits memory usage through the use of a sliding window.

6 Discussion, conclusions and future work

In the current work, we expanded the formalism of hypothetical answers to continuous queries with the possibility of communication delays, and showed how we could define a declarative semantics and a corresponding operational semantics by suitably adapting and enriching the definitions from [17].

Our work deals with communication delays without requiring any previous processing of the input stream, as long as bounds for delays are known. To the best of our knowledge, this is the first work providing this flexibility in the

context of a logical approach to stream query answering. This motivated us to develop a resolution search strategy driven by the arrival of data, instead of a static one. It would be interesting to try to combine our resolution strategy with dynamic strategies [25], or techniques to produce compact representations of search trees [41].

The algorithm we propose needs to store significant amounts of information. This was already a problem in [17], and the addition of delays enhances it (since invalid answers are in general discarded later). One possible way to overcome this limitation would be to assign confidence levels to schematic supported answers, and either discard answers when their confidence level is below a given threshold, or discard the answers with lowest confidence when there are too many. This requires having information about (i) the probability that a given premise is true, and (ii) the probability that that premise arrives with a given delay. These probabilities can be used to infer the likelihood that a given schematic supported answer will be confirmed. The relevant probabilities could be either evaluated by experts, or inferred using machine-learning tools.

Our approach extends naturally extends to the treatment of lossy channels. If we have a sub-probability distribution for communication delays (i.e. where the sum of all probabilities is stricly smaller than 1), then we are also allowing for the chance that information is lost in transit. Thus, the same analysis would also be able to estimate the likelihood that a given substitution is an answer, even though some of the relevant premises are missing. We plan to extend our framework along these lines, in order to have enough ingredients to develop a prototype of our system and conduct a full practical evaluation.

The presentation in [17] also discussed briefly how to add negation to the language. Those ideas are incompatible with the way we deal with communication delays, since they relied heavily on the fact that the exact time at which facts arrive is known. Adding negation to the current framework is also a direction for future research.

Acknowledgements. This work was partially supported by FCT through the LASIGE Research Unit, ref. UIDB/00408/2020 and ref. UIDP/00408/2020.

References

1. 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA. AAAI Press (2019)
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
3. Akidau, T., Balikov, A., Bekiroglu, K., Chernyak, S., Haberman, J., Lax, R., McVeety, S., Mills, D., Nordstrom, P., Whittle, S.: Millwheel: Fault-tolerant stream processing at internet scale. Proc. VLDB Endow. **6**(11), 1033–1044 (2013)
4. Alferes, J.J., Pereira, L.M.: Reasoning with Logic Programming, LNCS, vol. 1111. Springer (1996)
5. Alirezaie, M., Loutfi, A.: Automated reasoning using abduction for interpretation of medical signals. J. Biomed. Semant. **5**, 35 (2014)

6. Alur, R., Henzinger, T.A.: Real-time logics: Complexity and expressiveness. Inf. Comput. **104**(1), 35–77 (1993)
7. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Popa, L., Abiteboul, S., Kolaitis, P.G. (eds.) Procs. PODS. pp. 1–16. ACM (2002)
8. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) Procs. ESWC, Part I. LNCS, vol. 6088, pp. 1–15. Springer (2010)
9. Beck, H., Dao-Tran, M., Eiter, T.: Answer update for rule-based stream reasoning. In: Yang, Q., Wooldridge, M.J. (eds.) Procs. IJCAI. pp. 2741–2747. AAAI Press (2015)
10. Beck, H., Dao-Tran, M., Eiter, T., Fink, M.: LARS: A logic-based framework for analyzing reasoning over streams. In: Bonet and Koenig [12], pp. 1431–1438
11. Ben-Ari, M.: Mathematical Logic for Computer Science. Springer, 3rd edn. (2012)
12. Bonet, B., Koenig, S. (eds.): 29th AAAI Conference on Artificial Intelligence, AAAI 2015, Austin, TX, USA. AAAI Press (2015)
13. Brandt, S., Kalayci, E.G., Ryzhikov, V., Xiao, G., Zakharyaschev, M.: Querying log data with metric temporal logic. J. Artif. Intell. Res. **62**, 829–877 (2018)
14. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about datalog (and never dared to ask). IEEE Trans. Knowl. Data Eng. **1**(1), 146–166 (1989)
15. Chomicki, J., Imielinski, T.: Temporal deductive databases and infinite objects. In: Edmondson-Yurkanan, C., Yannakakis, M. (eds.) Procs. PODS. pp. 61–73. ACM (1988)
16. Cruz-Filipe, L., Gaspar, G., Nunes, I.: Hypothetical answers to continuous queries over data streams. CoRR **abs/1905.09610** (2019)
17. Cruz-Filipe, L., Gaspar, G., Nunes, I.: Hypothetical answers to continuous queries over data streams. In: Procs. AAAI. pp. 2798–2805. AAAI Press (2020)
18. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. **44**(3), 15:1–15:62 (2012)
19. Dao-Tran, M., Eiter, T.: Streaming multi-context systems. In: Sierra, C. (ed.) Procs. IJCAI. pp. 1000–1007. ijcai.org (2017)
20. Dell'Aglio, D., Valle, E.D., van Harmelen, F., Bernstein, A.: Stream reasoning: A survey and outlook. Data Sci. **1**(1–2), 59–83 (2017)
21. Denecker, M., Kakas, A.C.: Abduction in logic programming. In: Kakas, A.C., Sadri, F. (eds.) Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski. LNCS, vol. 2407, pp. 402–436. Springer (2002)
22. Dung, P.M.: Negations as hypotheses: An abductive foundation for logic programming. In: Furukawa, K. (ed.) Procs. ICLP. pp. 3–17. MIT Press (1991)
23. Ghanem, T.M., Hammad, M.A., Mokbel, M.F., Aref, W.G., Elmagarmid, A.K.: Incremental evaluation of sliding-window queries over data streams. IEEE Trans. Knowl. Data Eng. **19**(1), 57–72 (2007)
24. Gray, A.J., Nutt, W., Williams, M.H.: Answering queries over incomplete data stream histories. IJWIS **3**(1/2), 41–60 (2007)
25. Guo, H., Gupta, G.: Dynamic reordering of alternatives for definite logic programs. Comput. Lang. Syst. Struct. **35**(3), 252–265 (2009)
26. Gupta, A., Mumick, I.S., Subrahmanian, V.: Maintaining views incrementally. In: Buneman, P., Jajodia, S. (eds.) Procs. SIGMOD. pp. 157–166. ACM Press (1993)
27. Hu, P., Motik, B., Horrocks, I.: Optimised maintenance of datalog materialisations. In: McIlraith and Weinberger [38], pp. 1871–1879

28. Inoue, K.: Hypothetical reasoning in logic programs. J. Log. Program. **18**(3), 191–227 (1994)
29. Isah, H., Abughofa, T., Mahfuz, S., Ajerla, D., Zulkernine, F.H., Khan, S.: A survey of distributed data stream processing frameworks. IEEE Access **7**, 154300–154316 (2019)
30. Jr., W.L.: On semantic issues connected with incomplete information databases. ACM Trans. Database Syst. **4**(3), 262–296 (1979)
31. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems **2**(4), 255–299 (1990)
32. Lang, W., Nehme, R.V., Robinson, E., Naughton, J.F.: Partial results in database systems. In: Dyreson, C.E., Li, F., Özsu, M.T. (eds.) Procs. SIGMOD. pp. 1275–1286. ACM (2014)
33. de Leng, D., Heintz, F.: Approximate stream reasoning with metric temporal logic under uncertainty. In: AAAI2019 [1], pp. 2760–2767
34. Li, J., Tufte, K., Shkapenyuk, V., Papadimos, V., Johnson, T., Maier, D.: Out-of-order processing: a new architecture for high-performance stream systems. Proc. VLDB Endow. **1**(1), 274–288 (2008)
35. Li, M., Liu, M., Ding, L., Rundensteiner, E.A., Mani, M.: Event stream processing with out-of-order data arrival. In: Procs. ICDCS. p. 67. IEEE Computer Society (2007)
36. Libkin, L.: Incomplete data: what went wrong, and how to fix it. In: Hull, R., Grohe, M. (eds.) Procs. PODS. pp. 1–13. ACM (2014)
37. Lloyd, J.W.: Foundations of Logic Programming. Springer (1984)
38. McIlraith, S.A., Weinberger, K.Q. (eds.): 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, LA, USA. AAAI Press (2018)
39. Meadows, B.L., Langley, P., Emery, M.J.: Seeing beyond shadows: Incremental abductive reasoning for plan understanding. In: Plan, Activity, and Intent Recognition. AAAI Workshops, vol. WS-13-13. AAAI (2013)
40. Motik, B., Nenov, Y., Piro, R.E.F., Horrocks, I.: Incremental update of datalog materialisation: the backward/forward algorithm. In: Bonet and Koenig [12], pp. 1560–1568
41. Nishida, N., Vidal, G.: A framework for computing finite SLD trees. J. Log. Algebraic Methods Program. **84**(2), 197–217 (2015)
42. Özçep, Ö.L., Möller, R., Neuenstadt, C.: A stream-temporal query language for ontology based data access. In: Lutz, C., Thielscher, M. (eds.) Procs. KI. LNCS, vol. 8736, pp. 183–194. Springer (2014)
43. Razniewski, S., Korn, F., Nutt, W., Srivastava, D.: Identifying the extent of completeness of query answers over partially complete databases. In: Sellis, T.K., Davidson, S.B., Ives, Z.G. (eds.) Procs. SIGMOD. pp. 561–576. ACM (2015)
44. Reiter, R.: Towards a logical reconstruction of relational database theory. In: Brodie, M.L., Mylopoulos, J., Schmidt, J.W. (eds.) Procs. Intervale. pp. 191–233. Topics in information systems, Springer (1984)
45. Ronca, A., Kaminski, M., Grau, B.C., Motik, B., Horrocks, I.: Stream reasoning in temporal datalog. In: McIlraith and Weinberger [38], pp. 1941–1948
46. Stonebraker, M., Çetintemel, U., Zdonik, S.B.: The 8 requirements of real-time stream processing. SIGMOD Record **34**(4), 42–47 (2005)
47. Valle, E.D., Ceri, S., van Harmelen, F., Fensel, D.: It's a streaming world! Reasoning upon rapidly changing information. IEEE Intelligent Systems **24**(6), 83–89 (2009)
48. Walega, P.A., Kaminski, M., Grau, B.C.: Reasoning over streaming data in metric temporal datalog. In: AAAI2019 [1], pp. 3092–3099

49. Zaniolo, C.: Logical foundations of continuous query languages for data streams. In: Barceló, P., Pichler, R. (eds.) Procs. Datalog 2.0. LNCS, vol. 7494, pp. 177–189. Springer (2012)