Can't you answer while you wait?

Luís Cruz-Filipe[1*], Graça Gaspar[2] and Isabel Nunes[2,3]

[1*]Dept. Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark.
[2]LASIGE, Faculty of Sciences, University of Lisbon, Campo Grande, 1749-001 Lisboa, Portugal.
[3]Dept. Informatics, Faculty of Sciences, University of Lisbon, Campo Grande, 1749-001 Lisboa, Portugal.

*Corresponding author(s). E-mail(s): lcfilipe@gmail.com;
Contributing authors: mdgaspar@ciencias.ulisboa.pt;
minunes@ciencias.ulisboa.pt;

Abstract

Many modern-day systems rely on information that is constantly arriving, and they need to make decisions based on it – a problem known as *continuous query answering*. In many situations, these systems can benefit from identifying possible outcomes that are consistent with the data available so far. Such scenarios are called *hypothetical answers*, and previous work has defined them precisely and shown how they can be updated in step with the arrival of additional input.
Most existing formalisms assume that data always arrives instantaneously, which is not realistic. In this work, we relax this problem by allowing data to arrive later, and potentially out of order. By revisiting the underlying intuitions in previous work, we develop a more general framework that supports communication delays. The interaction between communication delays and negation poses some challenging problems, which we address using fixpoint theory. We show that the relevant fixpoints can be computed in finite time by a carefully designed algorithm.

Keywords: continuous query answering, Temporal Datalog, negation-as-failure, communication delays

1 Introduction

Many systems in current use make decisions based on information that is constantly arriving, e.g. from sensors that monitor their state. These systems must reason over this continuous influx of information, which is usually called a *data stream*. The task of continuously reasoning over a data stream is called *continuous query answering*.

An important line of work [11, 23, 49, 52, 57] models continuous queries as logic programs where the set of facts arrives through a data stream, and computes answers to queries by logic-based methods. One practical limitation is that the information required to compute a specific answer to a query may arrive over a long period of time. This concern was the motivation behind the introduction of *hypothetical answers* [17], which provide information on answers to the query that may be produced in the future. Hypothetical answers take into account the available data, and indicate which facts the data stream must produce in order for them to become "real" answers. They can therefore be used not only to foresee possible answers to the query, but also to implement measures to prevent these answers from materializing (e.g. in a scenario where answers to the query correspond to system malfunctions).

Another relevant practical concern that must be addressed is that communication is in general not instantaneous – information may be delayed, and the data produced by the data stream is not necessarily in chronological order. From a theoretical point of view, this can be dealt with by forcing the data from the data stream to be ordered [7, 24, 53]: information with timestamp greater than t is "put on hold" until all data about time point t is produced. However, this requires knowledge about which information will still arrive, which is not readily available in practice.

If we allow the data stream to produce unordered data, the typical strategies for continuous query answering no longer apply, as they rely on grouping information by time point before processing it. 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. Our only assumption is that there is a known limit 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.

Our procedure generalizes ideas from previous work [20] for positive programs, but the intuitive strategy used in that work to address communication delays directly conflicts with the treatment of negation. This is due to the fact that default negation is non-monotonic, while hypothetical answers are by nature monotonic – making their interplay complex to handle. We address this conflict by using ideas from logic programming. Intuitively, we work with Kleene's 3-valued logic, where answers to queries may be known to be true, known to be false, or unknown (as of yet), and we define a formal operational semantics as the least fixpoint of a monotonic operator over a suitably defined bilattice [28].

Contribution

We define a notion of generalized hypothetical answers to continuous queries in the presence of communication delays for the whole language of Temporal Datalog with negation that we introduced in previous work [20]. We introduce an operational semantics for these programs that precisely matches this notion, using an appropriate *negation update operator* that acts on an *evidence lattice* whose elements are approximations of generalized hypothetical answers, in the spirit of Approximation Fixpoint Theory [26]. We show that this operator has a unique fixpoint, making the semantics well-defined. Finally, we introduce an algorithm for computing hypothetical answers that avoids potentially infinite computations by lazily instantiating variables only when needed (using the notion of *local most general unifier* and computing fixpoints of the negation update operator in the "right" order).

Publication history

This article combines results originally presented in conference publications [18] and [19]. The presentation has been thoroughly revised for consistency and conciseness. We have added some proofs that were missing in the original presentations, as well as new results on soundness and completeness of the different algorithms. We have rephrased some of the most important results in Section 5, namely Theorem 1, and gave a more precise characterization of the correspondences between the notions of generalized hypothetical answer, SLDNF-refutation with future premises, and schematic hypothetical answer. In particular, the presentation has been simplified by the introduction of a definition of the set $\mathcal{S}_\tau^{\mathrm{SLDNF}}$ (Definition 8) and its characterization in the new Theorem 3.

The constructions for negation are now also illustrated by examples, which were not included in the original publication. Finally, we have expanded the discussion on complexity, including a more precise analysis of both the program complexity and data complexity of the online step in our algorithm, and arguing from these for the practical applicability of our algorithm.

Structure

In Section 2, we review the syntax and semantics of Temporal Datalog with negation, as well as the original framework for hypothetical query answering. Hypothetical answers in the presence of communication delays are defined in Section 3, and Section 4 defines the operational semantics and shows that it coincides with the declarative definitions. Section 5 presents a recursive definition of the (infinite) set of all hypothetical answers at a given point in time, and Section 6 shows an algorithm for computing these sets symbolically, and argues for its correctness. Related work is discussed in Section 7, and we conclude in Section 8.

2 Background

We work in the framework of *Temporal Datalog with negation*, which we introduced earlier [20], and which corresponds to extending Datalog$^\neg$ with a special temporal sort originally described by Chomicki and Imielinski [16]. The formalism for writing continuous queries over datastreams is adapted from the proposal of Ronca et al. [52], and the framework for computing hypothetical answers to these queries builds on ideas from abduction in logic programming [34] and a generalization of SLD-resolution [44].

Syntax of Temporal Datalog with negation

A *signature* for Temporal Datalog [15] consists of a set of constants, a set of variables and a set of predicate symbols. Constants and variables have one of two sorts: *object* or *temporal*, and sorting extends to terms: 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: such predicates can be extended with an extra dummy temporal parameter which always takes the value 0.) Atoms are built from applying predicates to terms in the standard way, and a literal is either an atom (positive literal) or its negation (negative literal). If A is a set of literals, we write A^+ for the subset of positive literals in A, and A^- for the set of atoms that appear negated in A.

A rule is a disjunction of literals with at least one positive literal, and a program is a finite set of rules. Typically rules are written in the form $\ell_1, \ldots, \ell_n \to P$ where each ℓ_i is a literal and P is an atom. We call $\ell_1, \ldots, \ell_n$ the *body* of the rule and P the *head* of the rule. A program whose rules do not include negative literals is called *positive*.

We write $\mathsf{var}(P)$ for the set of variables occurring in an atom P, and extend this function homomorphically to rules and sets. An atom, literal or rule is *ground* if it contains no variables (i.e. all terms are built from only constants and temporal operations). A *fact* is a rule with empty body ($n = 0$); it follows from the previous requirement that all facts are ground.

Following the terminology from deductive databases, we call a predicate symbol *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.

A substitution is a function θ mapping variables to terms of the expected sort, such that $\theta(X) \neq X$ for only finitely many X. The set of variables for which this holds is called the *support* of θ, denoted $\mathsf{supp}(\theta)$. Given a rule r, 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 *closed* substitution is a substitution that maps all variables to ground terms.

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.

We illustrate these concepts with an example adapted from previous work on continuous query answering [52, Example 1], which we use as running example throughout this article.

Example 1.

The following program Π_{WT} tracks activation of cooling measures in a set of wind turbines equipped with sensors, based on temperature readings $\mathsf{Hot}(Device, Time)$ that arrive instantly through the datastream. The intensional predicates are defined by the following rules.

$$\begin{aligned}
\mathsf{Hot}(X,T) &\to \mathsf{Flag}(X,T)\\
\mathsf{Flag}(X,T) \wedge \mathsf{Flag}(X,T+1) &\to \mathsf{Cool}(X,T+1)\\
\mathsf{Cool}(X,T) \wedge \mathsf{Flag}(X,T+1) &\to \mathsf{Shdn}(X,T+1)\\
\neg\mathsf{Shdn}(X,T) &\to \mathsf{OK}(X,T-1)
\end{aligned}$$

If a sensor reports a high temperature in a given turbine, then that turbine is flagged. If a turbine is flagged twice in a row, then a cooling system should be activated. If the turbine is still flagged after this system is activated, then it shuts down. The absence of a shutdown at a given point in time means that the turbine was working properly in the preceding time point (either not heating up, or being properly cooled down).

Shutdowns are tracked by the query $Q_S = \langle \mathsf{Shdn(X,T)}, \Pi_{WT}\rangle$. ◁

Semantics of Temporal Datalog with negation

The semantics of Temporal Datalog is defined over Herbrand models [44], evaluating time terms to a natural number in the obvious way. As usual with Datalog-based languages that include negation, we only define the semantics for programs that obey a notion of *stratification* [17].

A stratification of a Datalog program Π [37] is an infinite sequence of disjoint programs $\Pi_0, \ldots, \Pi_n, \ldots$ such that $\Pi = \cup_{k\in\mathbb{N}}\Pi_k$ and, for each predicate symbol p, (i) the definition of p is contained in one Π_k, (ii) if p occurs positively in the body of a rule with head q, then the definition of q is contained in Π_i for some $i \geq k$, and (iii) if p occurs negatively in the body of a rule with head q, then the definition of q is contained in Π_i for some $i > k$. Π is stratifiable if it admits a stratification.

Definition 1.

If Π is a program in Temporal Datalog, then its temporal closure *is the (Datalog) program $\Pi^{\downarrow}$ defined as follows:*

- *for each $(n+1)$-ary predicate symbol p in the signature underlying Π, the signature for $\Pi^{\downarrow}$ contains a family of n-ary predicate symbols $\{p_t\}_{t\in\mathbb{N}}$;*
- *for each rule in Π, $\Pi^{\downarrow}$ contains all rules obtained by instantiating its temporal parameter in all possible ways and replacing $p(x_1, \ldots, x_n, t)$ by $p_t(x_1, \ldots, x_n)$.*

$\Pi^{\downarrow}$ is T-stratifiable if Π is stratifiable.

Note that $\Pi^{\downarrow}$ is, in general, an infinite program, but T-stratification is decidable as long as Π is finite [20].

Given an interpretation I, satisfaction of rules and programs is defined as usual in Datalog, taking into account that negation is safe. From I we can generate an interpretation $I^{\downarrow}$ over the signature for $\Pi^{\downarrow}$ in the obvious way. Since I is a model of

Π iff $I^{\downarrow}$ is a model of $\Pi^{\downarrow}$, it follows that T-stratified programs have a unique answer set that coincides with their well-founded model. Given an atom P, we write $\Pi \models P$ if P is true in the well-founded model for Π.

SLD-resolution

Entailment can be computed by SLD-resolution. We summarize the key relevant concepts and results, which can be found in standard reference texts [12, 44].

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 $\wedge_i \alpha_i \to \alpha$, 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 $\wedge_i \alpha_i \to \alpha$, 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 Π be a positive program and G be a goal. An *SLD-derivation* of $\Pi \cup \{G\}$ is a sequence $G_0, G_1, \ldots$ of goals, a sequence $C_1, C_2, \ldots$ of α-renamings[1] of program clauses of Π 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 $\Pi \cup \{G\}$ is a finite SLD-derivation of $\Pi \cup \{G\}$ 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.

To deal with negative literals in the bodies of rules, SLD-resolution is extended with *negation-as-failure* (NF). SLDNF-derivations are allowed to include steps where one ground negative literal $\neg P$ is removed from a goal provided that there is an SLDNF-refutation for $\Pi \cup \{P\}$.

SLDNF-resolution is sound and complete for stratified programs. If θ is a computed answer for $\Pi \cup \{G\}$, then $\Pi \models \neg(\forall G\theta)$, where $\forall G\theta$ denotes the formula obtained by universally quantifying over all free variables in $G\theta$. Conversely, if $\Pi \models \neg(\forall G\theta)$, then there exist σ and γ such that $\theta = \sigma\gamma$ and σ is a computed answer for $\Pi \cup \{G\}$. 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 SLDNF-refutation.

Temporal queries over datastreams

To consider temporal queries, we assume that the set of rules in a program is known from the start, but the facts only become available as time progresses. We model this via the notion of *dataset*. A dataset is a family $D = \{D|_\tau \mid \tau \in \mathbb{N}\}$, where $D|_\tau$ is meant to represent the set of EDB facts delivered by a data stream at time point τ. As such, we assume 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.

[1]Two formulas are said to be α-renamings of each other if they only differ in the names of their bound variables [9].

By convention, $D_{-1} = \emptyset$, and we refer to *time point* -1 to denote the point of time before the data stream starts delivering facts. We also abuse notation, and often write simply D instead of $\bigcup D$.

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 $\mathbb{A}(Q, D, \tau)$.

Example 2.

Consider again the example program Π_{WT} *and query* Q_S *from Example* 1*. If we assume* $D_0 = \{\mathsf{Hot}(\mathsf{wt1}, 0)\}$*, then at time point* 0 *there is no answer to* Q_S*. If* $\mathsf{Hot}(\mathsf{wt1}, 1)$ *arrives to* D *at time point* 1*, then* $D_1 = D_0 \cup \{\mathsf{Hot}(\mathsf{wt1}, 1)\}$*, and there still is no answer to* Q_S*. Finally, the arrival of* $\mathsf{Hot}(\mathsf{wt1}, 2)$ *to* D *at time point* 2 *yields* $D_2 = D_1 \cup \{\mathsf{Hot}(\mathsf{wt1}, 2)\}$*, allowing us to infer* $\mathsf{Shdn}(\mathsf{wt1}, 2)$*. Then* $\{X := \mathsf{wt1}, T := 2\} \in \mathbb{A}(Q_S, D, 2)$*.*

Proper functioning of the turbines' cooling systems can be tracked by the query $Q_{OK} = \langle \mathsf{OK}(X, T), \Pi_{WT} \rangle$*. Suppose that* $\mathsf{wt2}$ *is also a constant in our language. Since readings from the sensor arrive instantly, we know that* $\mathsf{Hot}(\mathsf{wt2}, 0) \notin D$*, and therefore* $\Pi_{WT} \cup D \not\models \mathsf{Shdn}(\mathsf{wt2}, 2)$*, from which* $\Pi_{WT} \cup D \models \mathsf{OK}(\mathsf{wt2}, 1)$*. Therefore* $\{X := \mathsf{wt2}, T := 1\} \in \mathbb{A}(Q_{OK}, D, 0)$*.* $\triangleleft$

The second part of this example relies heavily on facts being delivered instantly by the data stream. In the presence of communication delays, the situation becomes significantly more complex.

Hypothetical query answering

In Example 1, the answer to the query Q_S could only be determined when $\tau = 2$. However, we could already infer that this answer might arise from the fact that $D_0 = \{\mathsf{Hot}(\mathsf{wt1}, 0)\}$. The later delivery of $\mathsf{Hot}(\mathsf{wt1}, 1)$ supports this possibility, while its absence from the data stream would have discarded it.

The formalism of *hypothetical answers* [20, 17] builds on this idea. We call a ground atom $p(t_1, \ldots, t_n)$ *future-possible for* τ if $\tau < t_n$.

Definition 2.

Let $D = \{D|_\tau \mid \tau \in \mathbb{N}\}$ *be a dataset,* τ' *be a time point and* H *be a consistent finite set of ground EDB literals. A dataset* $D' = \{D'|_\tau \mid \tau \in \mathbb{N}\}$ *is a* possible evolution *of* D *at time* τ'*, denoted* $D' \supseteq_{\tau'} D$*, if* $D'|_\tau = D|_\tau$ *for all* $\tau \leq \tau'$*.*

A possible evolution D' *of* D *at time* τ' *is* compatible *with* H *if: (i)* $H^+ \subseteq D'$*; and (ii)* $H^- \cap D' = \emptyset$*.*

Recall that H^+ (H^-) is the set of atoms that occur (negated) in H. Intuitively, H^+ contains the facts that *must* be produced by the data stream, and H^- the facts that *cannot* be produced by the data stream – so these two sets should be disjoint, which is ensured by requiring H to be consistent.

Definition 3.

A hypothetical answer *to a query $Q = \langle P, \Pi \rangle$ over D_τ is a pair $\langle \theta, H \rangle$, where θ is a substitution and H is a consistent finite set of ground EDB literals such that: (i) $\mathsf{supp}(\theta) = \mathsf{var}(P)$; (ii) H^+ and H^- only contain atoms future-possible for τ; (iii) $\Pi \cup D' \models P\theta$ for each possible evolution D' of D at time τ compatible with H; and (iv) H is minimal with respect to set inclusion.*

We write $\mathbb{H}(Q, D, \tau)$ for the set of all hypothetical answers to Q over D_τ.

Intuitively, a hypothetical answer to Q indicates a substitution that will become an answer to the query if the data stream evolves in the way described by the hypotheses.

Definition 4.

Let $Q = \langle P, \Pi \rangle$ be a query, D be a dataset and τ be a time point. A set of ground EDB literals E is evidence *supporting $\langle \theta, H \rangle \in \mathbb{H}(Q, D, \tau)$ if: (i) $E^+ \subseteq D_\tau$ and $E^- \cap D_\tau = \emptyset$; (ii) $E^+ \cup E^- \neq \emptyset$; (iii) $\Pi \cup D' \models P\theta$ for each possible evolution D' of D at time -1 compatible with $E \cup H$; and (iv) E is minimal with respect to set inclusion.*

A supported answer *to Q over D_τ is a triple $\langle \theta, H, E \rangle$ such that $\Pi \cup H \not\models P\theta$ and E is evidence supporting $\langle \theta, H \rangle$. We write $\mathbb{E}(Q, D, \tau)$ for the set of all supported answers to Q over D_τ.*

Intuitively, E^+ is the set of facts already produced by the data stream that are essential to the hypothetical answer, while E^- is the set of facts whose guaranteed absence is relevant for the hypothetical answer. This is captured by requiring compatibility at time -1 – we are effectively "forgetting" D and only requiring that D' contain the information from E^+ and H^+ and not include anything from E^- or D^-.

The requirement $\Pi \cup H \not\models P\theta$ captures the intuition that the answer θ is indeed dependent on some fact that can be deduced from the current state of the data stream.

Example 3.

In Example 1, $\langle [X := \mathsf{wt1}, T := 2], \{\mathsf{Hot}(\mathsf{wt1}, 2)\} \rangle$ is a hypothetical answer for Q_S and D at time 1. This answer is supported by the evidence $\{\mathsf{Hot}(\mathsf{wt1}, 0), \mathsf{Hot}(\mathsf{wt1}, 1)\}$.

The pair $\langle [X := \mathsf{wt1}, T := 5], \{\mathsf{Hot}(\mathsf{wt1}, t) \mid t = 3, 4, 5\} \rangle$ is also a hypothetical answer for Q_S and D at time 1, but it is not supported. ◁

Computing hypothetical answers

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 $\Pi \cup \{\neg P\}$ until it reaches a goal containing no positive IDB atoms (i.e. it may contain EDB atoms and negative IDB atoms), and returns a set $\mathcal{P}_Q$ containing a pair $\langle \theta, H \rangle$ for each successful derivation, where θ is the computed substitution for that derivation and H contains all the atoms in the leaf. For each negative literal $\neg p(t_1, \ldots, t_n)$, a fresh auxiliary query $\langle p(X_1, \ldots, X_n), \Pi \rangle$ is generated by replacing all terms with variables. All generated queries are then in turn pre-processed, and may spawn additional auxiliary queries. The process is iterated until no fresh queries arise.

Example 4.

Consider again the program from Example 1 and the query Q_{OK}. Pre-processing Q_{OK} yields $\mathcal{P}_{Q_{OK}} = \{\langle\emptyset, \{\neg\mathsf{Shdn}(X, T+1)\}\rangle\}$, which generates the auxiliary query Q_S also mentioned in Example 1. Pre-processing this query yields the singleton set

$$\mathcal{P}_{Q_S} = \{\langle\emptyset, \{\mathsf{Hot}(X, T-2), \mathsf{Hot}(X, T-1), \mathsf{Hot}(X, T)\}\rangle\}\,.$$

Since no fresh auxiliary queries are generated, the process terminates. $\triangleleft$

Pre-processing can be shown to terminate under some assumptions [17], which we do not discuss here. Soundness and completeness of pre-processing state that:

- if $\langle\theta, H\rangle \in \mathcal{P}_Q$, then there exist a dataset D and substitution σ such that $Q\theta\sigma$ is ground, $H\theta\sigma \subseteq D$ and $\theta\sigma$ is an answer to Q over Π and D;
- if σ is an answer to Q over Π and D, then there exists $\langle\theta, H\rangle \in \mathcal{P}_Q$ such that $\sigma = \theta\rho$ for some ρ and $H\sigma \subseteq D$.

For each time point τ and query Q, we recursively compute a set of *schematic hypothetical answers* $\mathcal{S}_\tau(Q)$ of the form $\langle\theta, E, H\rangle$, where θ is a substitution and E and H are sets of (not necessarily ground) literals not containing any IDB atoms. The reason for working with schematic answers is to avoid the combinatorial explosion that arises by considering all possible instantiations of all variables – instead we delay instantiation until the data stream provides values for the variables.

The set $\mathcal{S}_\tau(Q)$ is computed by (i) unifying the hypotheses in each element of $\mathcal{P}_Q$ with $D|_\tau$; (ii) copying the schematic hypothetical answers in $\mathcal{S}_{\tau-1}(Q)$ for which any elements that should arrive in $D|_\tau$ are indeed there (this may lead to further instantiation of variables); and (iii) recursively checking $S_\tau(Q)$ to determine whether the truth value of any negative literals among the current hypotheses can be established (this may also lead to further instantiation of variables).

We do not include the details here, as the algorithm is complex and generalized by the one that we present later on. Schematic hypothetical answers represent sets of hypothetical answers that can be obtained by grounding and recursively replacing negative IDB literals using the information in step (iii). (This correspondence is mostly of theoretical interest, though, since in practice schematic answers are easier to read and understand.)

The following example illustrates the general intuitions.

Example 5.

Consider again the scenario of Example 1. Since $D_0 = \{\mathsf{Hot}(\mathsf{wt1}, 0)\}$, $\mathcal{S}_0(Q_S)$ is a singleton containing the tuple

$$\{\langle[X := \mathsf{wt1}, T := 2]\,, \{\mathsf{Hot}(\mathsf{wt1}, 0)\}, \{\mathsf{Hot}(\mathsf{wt1}, i) \mid i = 1, 2\}\rangle\}\,.$$

From $\mathcal{P}_{Q_{OK}}$, we also include $\langle [T := 1], \emptyset, \{\neg\mathsf{Shdn}(X, 2)\}\rangle\}$ in $\mathcal{S}_0(Q_{OK})$.

Since there is no hypothetical answer that can yield $\mathsf{Shdn}(\mathsf{wt2}, 2)$*, this last set can be updated to*

$$\{\langle [X := \mathsf{wt2}, T := 1], \{\neg\mathsf{Shdn}(\mathsf{wt2}, 2)\}, \emptyset\rangle, \langle [X = \mathsf{wt1}, T := 1], \emptyset, \{\neg\mathsf{Shdn}(\mathsf{wt1}, 2)\}\rangle\}\,.$$

After D_1 and D_2 are available, we can compute

$$\begin{aligned}
\mathcal{S}_1(Q_S) = \{&\langle [X := \mathsf{wt1}, T := 2], \{\mathsf{Hot}(\mathsf{wt1}, i) \mid i = 0, 1\}, \{\mathsf{Hot}(\mathsf{wt1}, 2)\}\rangle,\\
&\langle [X := \mathsf{wt1}, T := 3], \{\mathsf{Hot}(\mathsf{wt1}, 1)\}, \{\mathsf{Hot}(\mathsf{wt1}, i) \mid i = 2, 3\}\rangle\}\\
\mathcal{S}_2(Q_S) = \{&\langle [X := \mathsf{wt1}, T := 2], \{\mathsf{Hot}(\mathsf{wt1}, i) \mid i = 0, 1, 2\}, \emptyset\rangle,\\
&\{\langle [X := \mathsf{wt1}, T := 3], \{\mathsf{Hot}(\mathsf{wt1}, i) \mid i = 1, 2\}, \{\mathsf{Hot}(\mathsf{wt1}, 3)\}\rangle,\\
&\langle [X := \mathsf{wt1}, T := 4], \{\mathsf{Hot}(\mathsf{wt1}, 2)\}, \{\mathsf{Hot}(\mathsf{wt1}, i) \mid i = 3, 4\}\rangle\}
\end{aligned}$$

At this point, there is an answer that proves $\mathsf{Shdn}(\mathsf{wt1}, 2)$*, and the hypothetical answer* $\langle [X = \mathsf{wt1}, T := 1], \emptyset, \{\neg\mathsf{Shdn}(\mathsf{wt1}, 2)\}\rangle$ *is removed from $\mathcal{S}_2(Q_{OK})$.* ◁

3 Hypothetical answers with communication delays

The formalism described in the previous section assumes that facts are received instantaneously at the data stream, in the sense that facts with timestamp τ appear in $D|_\tau$, if at all. This assumption is essential for discarding hypothetical answers that are no longer possible, and for the treatment of negation.

In practice, however, communications take time, meaning that facts with timestamp τ may arrive at the datastream at a later point in time. The purpose of this work is to extend the formalism of hypothetical answers to accommodate for these *communication delays*. Our working assumptions are 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, making this a reasonable requirement.

While it makes sense to assume that a bound on communication delays is known, this bound may be different for different predicates, or even for different instantiations of the same predicate – in Example 1, it could be the case that sensor reports from turbine wt1 always arrive within two time units, but those from turbine wt2 can take up to five time units (for example, due to geographical location). Our only restriction is that the bound on the communication delay may not depend on the timestamp. 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.

We formalize this requirement by assuming a function δ mapping each ground EDB atom in the language of Π to a natural number, with the restriction that: if α, β are atoms differing only in their temporal argument, then $\delta(\alpha) = \delta(\beta)$. 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 $\delta(p) = \delta(p(X_1, \ldots, X_n))$.

We start by generalizing the notion of future-possible atom to account for the possibility of communication delays. In general, we reuse terminology from the previous development whenever the new notions coincide with the old for the special case $\delta(p) = 0$ for all p (i.e. there are no communication delays).

Definition 5.

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

These are exactly the atoms that may still be delivered by the data stream after time instant τ, and as such the ones that it makes sense to include as (positive) hypotheses in hypothetical answers.

Possible evolutions, hypothetical answers and evidence are defined as before (Definitions 2 to 4), using this more general notion of future-possible atom.

Example 6.

We illustrate these concepts with the program from Example 1. We assume that $\delta(\mathsf{Hot}(\mathsf{wt1}, T)) = 2$ and $\delta(\mathsf{Hot}(\mathsf{wt2}, T)) = 5$, that $D|_0 = \{\mathsf{Hot}(\mathsf{wt1}, 0)\}$, and that $D|_1 = \emptyset$. Let $\theta = [X := \mathsf{wt1}, T := 2]$. Then

$$\langle \theta, \{\mathsf{Hot}(\mathsf{wt1}, 1), \mathsf{Hot}(\mathsf{wt1}, 2)\}\rangle \in \mathbb{H}(Q_S, D, 1)\,,$$

reflecting the intuition that $\mathsf{Hot}(\mathsf{wt1}, 1)$ *may still arrive in $D|_2$ or $D|_3$. This answer is supported by* $\mathsf{Hot}(\mathsf{wt1}, 0)$*.*

Likewise, if $\theta' = [X := \mathsf{wt2}, T := 2]$, then

$$\langle \theta', \{\mathsf{Hot}(\mathsf{wt2}, i) \mid i = 0, 1, 2\}\rangle \in \mathbb{H}(Q_S, D, 1)$$

and this hypothetical answer (or a similar one with fewer hypotheses) will also be in $\mathbb{H}(Q_S, D, \tau)$ for $\tau \leq 4$, since measurements from turbine wt2 *may take longer to arrive.* ◁

The key properties of hypothetical and supported answers still hold for this generalized notion. The proofs are adaptations of those for similar results in the setting without delays [20].

Proposition 1.

Let Q be a query, D be a dataset and τ be a time instant. If $\langle \theta, \emptyset\rangle \in \mathbb{H}(Q, D, \tau)$, then $\theta \in \mathbb{A}(Q, D, \tau)$.

Proof. Straightforward consequence of the definitions. □

Proposition 2.

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

Proof. Define a dataset D' by:

- $D'|_t = D|_t$, for $t \leq \tau$;
- $D'|_t$ contains the atoms in H^+ with timestamp t, for $t > \tau$.

Assume that $Q = \langle P, \Pi\rangle$. By construction, D' is a possible evolution of D at time τ compatible with H, therefore $\Pi \cup D' \models P\theta$ by definition of hypothetical answer. Since H is finite, taking τ' to be the highest timestamp in any atom in H it also follows that $\Pi \cup D'_{\tau'} \models P\theta$, and therefore $\theta \in \mathbb{A}(Q, D', \tau')$. □

4 Operational semantics for hypothetical answers

We define an operational semantics for hypothetical query answering in the presence of communication delays based on SLDNF-resolution, generalizing the ideas underlying the original work [20]. We develop the theory first in the setting where everything is ground, and delay working with variables until Section 6.

Given a program Π, we write Π^g for the grounded version of Π, i.e., the program containing all ground instances of all rules in Π.

Definition 6.

A (grounded) SLDNF-refutation with future premises *of $Q = \langle P, \Pi\rangle$ over D_τ with answer θ is a finite SLDNF-derivation of $\Pi^g \cup D_\tau \cup \{\neg P\theta\}$ whose last goal only contains:*

- *future-possible EDB atoms w.r.t. τ; or*
- *negated atoms for which the corresponding SLDNF-refutation with future premises does not end with the empty clause.*

Example 7.

Consider the setting of Example 6 and $\tau = 1$. We can build an SLDNF-refutation with future premises for Π_{WT} over D_1 with answer $[X := \mathsf{wt1}, T := 1]$ as follows:

$$\begin{tikzcd}
\leftarrow \mathsf{OK}(\mathsf{wt1}, 1) \arrow[d] & \leftarrow \mathsf{Shdn}(\mathsf{wt1}, 2) \arrow[d] \\
\leftarrow \neg\mathsf{Shdn}(\mathsf{wt1}, 2) & \vdots \arrow[d] \\
 & \leftarrow \mathsf{Hot}(\mathsf{wt1}, 1), \mathsf{Hot}(\mathsf{wt1}, 2)
\end{tikzcd}$$

The derivation for the original query (left) has one negated IDB atom in the last goal; the derivation for the atom in this goal (right, intermediate steps omitted) ends with two EDB atoms, both of which are future-possible – $\mathsf{Hot}(\mathsf{wt1}, 1)$ *because its timestamp is 1 and its delay is 2, and $\tau = 1 < 1 + 2$; and* $\mathsf{Hot}(\mathsf{wt1}, 2)$ *even has a timestamp* *higher than τ.* $\triangleleft$

Computed answers with premises are an operational counterpart to hypothetical answers.

Theorem 1.

Let Q be a query, D be a dataset and τ be a time instant. If $\langle\theta, H\rangle \in \mathbb{H}(Q, D, \tau)$, then there is an SLDNF-refutation with future premises for Q over D_τ with answer θ.

Proof. Assume that $Q = \langle P, \Pi\rangle$ and define a dataset D' as in the proof of Proposition 2. In particular, $D_\tau = D'_\tau$ and $\theta \in \mathbb{A}(Q, D', \tau')$ for some $\tau' \geq \tau$. By soundness and completeness of SLDNF-resolution for stratified programs, this is the case iff there is an SLDNF-refutation for $\Pi^g \cup D'_{\tau'} \cup \{\neg P\theta\}$.

Consider the SLDNF-derivation obtained by removing all steps in this refutation (including auxiliary derivations generated by processing negated literals) that unify literals in the goal with atoms from $D'_{\tau'} \setminus D_\tau$. This derivation is an SLDNF-refutation with future premises for Q over D_τ with answer θ: any atoms in its leaves must be in $D'_{\tau'} \setminus D_\tau = H^+$ by construction of D', and therefore are future-possible EDB atoms w.r.t. τ by definition of hypothetical answer, while any negated atoms that are not removed necessarily have a corresponding SLDNF-refutation that does not end with the empty clause. $\square$

The converse implication does not hold, which may be a bit surprising. We can build a hypothetical answer from an SLD-refutation with future premises by simply taking the future-possible atoms in the leaf of the refutation to be the set H. Then, following the stratification, we can build candidate hypothetical answers for the remaining derivations recursively. For each such derivation, there may be several of these, each containing:

- all the positive literals in the leaf of the derivation;
- for each negative literal $\neg\alpha$ in the leaf of the derivation, the negation of one element of each hypothetical answer to α.

The issue is that the candidate hypothetical answers may be inconsistent, as the following example shows.

Example 8.

Let p, q and r be binary predicate symbols and consider the following program Π_w.

$$\neg q(X,T), r(X,T) \rightarrow p(X,T)$$
$$r(X,T) \rightarrow q(X,T)$$

Consider the query $Q_w = \langle p(\mathsf{a}, 1), \Pi_w\rangle$ and assume that $\delta(r) > 0$. The following is an SLDNF-refutation with future premises for Q_w over $D_1 = \emptyset$ with answer $\emptyset$.

$$\begin{tikzcd}
\leftarrow p(\mathsf{a}, 1) \arrow[d] & \leftarrow q(\mathsf{a}, 1) \arrow[d] \\
\leftarrow \neg q(\mathsf{a}, 1), r(\mathsf{a}, 1) & \leftarrow r(\mathsf{a}, 1)
\end{tikzcd}$$

Indeed, the leaves in this derivation only contain negative literals or future-possible atoms; however, turning this into a successful SLDNF-refutation requires $r(\mathsf{a}, 1)$ to be both true (to complete the derivation on the left) and false (so that the derivation on the right fails). Trying to construct a hypothetical answer to Q_w in the way described above would yield $\{r(\mathsf{a}, 1), \neg r(\mathsf{a}, 1)\}$, which is inconsistent. ◁

Although the correspondence between the declarative and the operational semantics is not as tight as in the setting without communication delays [20], the existence of SLDNF-refutations with future premises that do not correspond to any hypothetical answer is not problematic: in later sections we show that we can still use SLDNF-refutations to compute hypothetical answers.

Theorem 2.

Let Q be a query, D be a dataset and τ be a time instant. If $\langle \theta, H, E\rangle \in \mathbb{E}(Q, D, \tau)$, then there is an SLDNF-refutation with future premises for Q over E^+ with answer θ.

Proof. Similar to the previous proof, but defining $D'|_t = D_\tau \cap E^+$ for $t \leq \tau$. Assume again that $Q = \langle P, \Pi\rangle$. Then D' is a possible evolution of D at time -1 compatible with $E \cup H$, and by definition of supported answer this again implies that $\Pi \cup D'_{\tau'} \models P\theta$ for some $\tau' > \tau$, and the argument above can be applied. □

The following properties are also important for the development below.

Proposition 3.

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

Proof. Assume that $\langle \theta, H\rangle \in \mathbb{H}(Q, D, \tau)$, and define $H^* = H_1 \cup H_2$ with:

- $H_1 = H^+ \cup (D_\tau \setminus D_{\tau'})$;
- H_2 contains all ground EDB atoms $p(t_1, \ldots, t_n, t)$ such that $\tau' < t + \delta(p(t_1, \ldots, t_n, t)) \leq \tau$ that are not in $D_\tau \setminus D_{\tau'}$.

H^* contains only atoms that are future-possible for τ', and since it is finite it has at least one minimal subset H' with respect to set inclusion that contains H^+ and satisfies condition (iii) of Definition 3. □

A similar reasoning yields the corresponding result for supported answers.

Proposition 4.

Let Q be a query, D be a dataset and τ be a time instant. If $\langle\theta, H, E\rangle \in \mathbb{E}(Q, D, \tau)$ and $\tau' < \tau$, then there exists $\langle\theta, H', E'\rangle \in \mathbb{E}(Q, D, \tau')$ such that $H^+ = (H')^+ \setminus (D_\tau \setminus D_{\tau'})$, $(E')^+ = E^+ \setminus (D_\tau \setminus D_{\tau'})$, and $H \cup E = H' \cup E'$.

5 Generalized hypothetical answers

The operational semantics in the previous section assumes that a hypothetical answer θ is known in advance. In practice, however, it is more interesting to compute all possible hypothetical answers as data arrives, preferably reusing previously computed information. In this section we take the first steps towards this goal. We define a more general notion of hypothetical answer that captures all SLDNF-refutations with future premises, and give a recursive characterization of the set of all such generalized hypothetical answers.

Throughout this section we assume a fixed query $Q_0 = \langle P_0, \Pi\rangle$ that has been preprocessed as described in Section 2, yielding a set of queries $\mathsf{Q} = \{Q_i\}_{i\in I}$ for some finite set of indices I. In particular, $Q_0 = Q_i$ for some $i \in I$, so properties that hold for all elements of Q also hold for Q_0.

Definition 7.

A generalized hypothetical answer *to Q over a τ-history D_τ is a family $\mathcal{S}$ of tuples $\langle P, \theta, E, H\rangle$ where:*

- *$\langle P, \Pi\rangle \in \mathsf{Q}$ is a query;*
- *θ is a closed substitution ranging over the variables in P;*
- *E and H are disjoint sets of ground literals containing only EDB atoms and negative literals, such that $E^+ \subseteq D_\tau$ and all elements of H^+ are future-possible w.r.t. τ;*
- *for every $D' \Supset_\tau D_\tau$, (i) $\Pi \cup D' \not\models E^-$ and (ii) if $\Pi \cup D' \models H$, then $\Pi \cup D' \models P\theta$.*

This notion differs from the hypothetical answers defined earlier (Definition 3) in several ways: it allows negated IDB atoms in hypotheses and evidence; it includes hypothetical answers for all queries in Q instead of only for the main one; and it includes several different hypothetical answers for each individual query. Furthermore, generalized hypothetical answers also generalize SLDNF-refutations with future premises.

Definition 8.

Let D_τ be a τ-history and assume that there is an SLDNF-refutation with future premises for Q_0 over D_τ. For each leaf of each derivation in this refutation, build a tuple $\langle P, \theta, E, H\rangle$ where:

- *$\neg P$ is the goal in the root of the derivation;*
- *θ is the answer computed by the derivation;*
- *$E = E_1 \cup E_2$ where E_1 contains the elements of D_τ that were unified with in some step of the branch of the derivation leading to the chosen leaf, and E_2 contains*

negated atoms that were removed in some step of the same branch of the derivation (because the corresponding SLDNF-refutation with future premises ends with the empty clause);

- *H contains all literals in the leaf of the derivation.*

We denote the set containing all these tuples by $\mathcal{S}_\tau^{\mathrm{SLDNF}}$.

Proposition 5.

In the setting of Definition 8, the set $\mathcal{S}_\tau^{\mathrm{SLDNF}}$ is a generalized hypothetical answer to Q over D_τ.

Proof. The first three points of Definition 7 hold by construction and the definition of SLDNF-refutation with future premises. The last point holds by soundness of SLDNF-resolution. □

The converse implication does not hold in general: if a generalized hypothetical answer $\mathcal{S}$ contains a tuple $\langle P, \theta, E, H\rangle$ where H^- contains some atom that can be proven, the last condition in Definition 7 is trivially satisfied, but it will not be possible to build an SLDNF-refutation for $\langle P, \Pi\rangle$ over D_τ.

Corollary 6.

Let $Q = \langle P, \Pi\rangle$ be a query. If $Q \in \mathsf{Q}$ and $\langle \theta, H, E\rangle \in \mathbb{E}(Q, D, \tau)$, then there is a generalized hypothetical answer $\mathcal{S}$ to Q over D_τ such that $\langle P, \theta, E, H\rangle \in \mathcal{S}$.

Proof. By combining Theorem 1 with Proposition 5. □

Proposition 7.

Let $\mathcal{S}$ be a generalized hypothetical answer such that $H = \emptyset$ for every $\langle P, \theta, E, H\rangle \in \mathcal{S}$. Then, for each such tuple, θ is an answer to the corresponding query $\langle P, \Pi\rangle$.

Proof. From the last property of Definition 7, taking $D' = D_\tau$, it immediately follows from (ii) that $\Pi \cup D \models P\theta$. □

Combining this proposition with the previous corollary, we regain the precise correspondence between answers to the original query(ies) and generalized hypothetical answers without hypotheses.

There are several advantages to working with generalized hypothetical answers: they are easier to understand, and they are also more compact – there may be several atoms in the leaves of auxiliary derivations, corresponding to several different ways in which that derivation can fail.

Example 9.

The SLDNF-refutation with future premises from Example 7 corresponds to the generalized hypothetical answer

$$\{\langle \mathsf{OK}(X,T), [X := \mathsf{wt1}, T := 1], \emptyset, \{\neg\mathsf{Shdn}(\mathsf{wt1}, 2)\}\rangle,$$

$$\langle \mathsf{Shdn}(X,T), [X := \mathsf{wt1}, T := 2], \{\mathsf{Hot}(\mathsf{wt1},0)\}, \{\mathsf{Hot}(\mathsf{wt1},i) \mid i = 1,2\}\rangle\} \,.$$

(The evidence $\mathsf{Hot}(\mathsf{wt1},0)$ *comes from* $D|_0$*, and is unified in one of the steps omitted in the derivation shown above.)*

This generalized hypothetical answer corresponds to two hypothetical answers for Q_{OK} *(both without evidence):* $\langle [X := \mathsf{wt1}, T := 1], \{\neg\mathsf{Hot}(\mathsf{wt1},1)\}\rangle$ *and* $\langle [X := \mathsf{wt1}, T := 1], \{\neg\mathsf{Hot}(\mathsf{wt1},2)\}\rangle$. ◁

Example 10.

The SLDNF-refutation with future premises from Example 8 also corresponds to a generalized hypothetical answer, namely

$$\{\langle p(\mathsf{a},1), \emptyset, \emptyset, \{\neg q(\mathsf{a},1), r(\mathsf{a},1)\}\rangle,$$
$$\langle q(\mathsf{a},1), \emptyset, \emptyset, \{r(\mathsf{a},1)\}\rangle\} \,.$$

The fact that $r(\mathsf{a},1)$ *must be both true and false in order to prove* $p(\mathsf{a},1)$ *is not taken into account, since Definition 7 does not require any relation between the different tuples in a generalized hypothetical answer.* ◁

We now define recursively a sequence $\{\mathcal{S}^{\downarrow}_n\}_{n\geq -1}$ such that $\mathcal{S}^{\downarrow}_i$ is a generalized hypothetical answer to Q over D_i containing exactly all the tuples that can be generated from an SLDNF-refutation with future premises.

Definition 9.

For every $Q \in \mathsf{Q}$ *and* $\langle \theta, H\rangle \in \mathcal{P}_Q$*,* $\mathcal{S}^{\downarrow}_{-1}$ *contains all tuples* $\langle P, \sigma|_Q, \emptyset, H\sigma\rangle$ *where* $Q = \langle P, \Pi\rangle$*,* σ *instantiates all free variables in* H *and* P*,* $\sigma = \theta\rho$ *for some* ρ*, and* $\sigma|_Q$ *is the restriction of* σ *to the variables that appear in* P*.*

The construction of $\mathcal{S}^{\downarrow}_{\tau+1}$ from $\mathcal{S}^{\downarrow}_{\tau}$ is done in two steps. First, we update $\mathcal{S}^{\downarrow}_{\tau}$ with the information from the dataset: we define an auxiliary generalized hypothetical answer $\mathcal{A}_{\tau+1}$ containing all tuples $\langle P, \theta, E \cup (H^+ \cap D|_{\tau+1}), H \setminus D|_{\tau+1}\rangle$ such that $\langle P, \theta, E, H\rangle \in \mathcal{S}^{\downarrow}_{\tau}$ and $H^+ \setminus D|_{\tau+1}$ only contains future-possible atoms w.r.t. $\tau + 1$.

The second step constructs $\mathcal{S}^{\downarrow}_{\tau+1}$ as the fixpoint of an operator that updates the sets of negative hypotheses and evidence in $\mathcal{A}_{\tau+1}$. This is the purpose of the next section: we define this operator over a suitably constructed lattice, and show that it is monotonic. We can then apply the Knaster-Tarski theorem [44] to conclude that it has a least fixpoint.

Below we fix τ and write simply $\mathcal{A}$ for $\mathcal{A}_{\tau+1}$.

The evidence lattice

We start by defining the *evidence lattice* for $\mathcal{A}$, which takes into account all ways in which elements of $\mathcal{A}$ can be updated with further knowledge. The set underlying the lattice is the set $\mathfrak{L}$ of all sets $\mathcal{X}$ of tuples $\langle P, \theta, E, H\rangle$ where E and H are disjoint and such that: (i) if $\langle P,\theta,E,H\rangle \in \mathcal{X}$, then there exists $\langle P,\theta,E',H'\rangle \in \mathcal{A}$ with $E \cup H = E' \cup H'$ and $E' \subseteq E$ and (ii) if $\langle P,\theta,E,H\rangle$ and $\langle P,\theta,E',H'\rangle$ are distinct elements of $\mathcal{X}$, then $E \cup H \neq E' \cup H'$.

Property (i) states that $\mathcal{X}$ corresponds to updating $\mathcal{A}$ with some learned evidence. Property (ii) states that this update is unique. By property (i), every element $\langle P,\theta,E,H\rangle \in \mathcal{X}$ corresponds to an element of $\mathcal{A}$ in a unique way; we refer to this element as the "element of $\mathcal{A}$ generating" $\langle P,\theta,E,H\rangle$.

Definition 10.

Let $\mathcal{X},\mathcal{Y} \in \mathfrak{L}$. We say that $\mathcal{X}$ is less proven *than $\mathcal{Y}$, denoted $\mathcal{X} \sqsubseteq \mathcal{Y}$, if for every tuple $\langle P,\theta,E,H\rangle \in \mathcal{X}$ there exists a tuple $\langle P,\theta,E',H'\rangle \in \mathcal{Y}$ such that $E \cup H = E' \cup H'$ and $E \subseteq E'$.*

Intuitively, $\mathcal{X} \sqsubseteq \mathcal{Y}$ represents that hypothetical answers in $\mathcal{Y}$ are "closer to being proven" than those in $\mathcal{X}$.

Lemma 8.

The relation $\sqsubseteq$ is a partial order over $\mathfrak{L}$.

Proof. Reflexivity and transitivity are straightforward.

For antisymmetry, choose $\mathcal{X},\mathcal{Y} \in \mathfrak{L}$ with $\mathcal{X} \sqsubseteq \mathcal{Y} \sqsubseteq \mathcal{X}$, and pick $\langle P,\theta,E,H\rangle \in \mathcal{X}$. Since $\mathcal{X} \sqsubseteq \mathcal{Y}$, there exists $\langle P,\theta,E',H'\rangle \in \mathcal{Y}$ such that $E \cup H = E' \cup H'$ and $E \subseteq E'$. But since also $\mathcal{Y} \sqsubseteq \mathcal{X}$, there must also exist $\langle P,\theta,E'',H''\rangle \in \mathcal{X}$ such that $E' \cup H' = E'' \cup H''$ and $E' \subseteq E''$.

It then follows that $E \cup H = E'' \cup H''$, which by (ii) implies that $E = E''$. Therefore $E \subseteq E' \subseteq E$, so $E = E'$, and by disjointness also $H = H'$. So $\langle P,\theta,E,H\rangle \in \mathcal{Y}$, whence $\mathcal{X} \subseteq \mathcal{Y}$.

A similar reasoning starting from a random element in $\mathcal{Y}$ establishes that $\mathcal{Y} \subseteq \mathcal{X}$, and therefore these two sets must be equal. □

Lemma 9.

Every subset of $\mathfrak{L}$ has a least upper bound w.r.t. $\sqsubseteq$.

Proof. Let $\mathfrak{D}$ be a subset of $\mathfrak{L}$ and define $\bigvee \mathfrak{D}$ as the set of all tuples $\langle P,\theta,E^\vee,H^\vee\rangle$ such that:

- $E^\vee = \bigcup\{E_i \mid \langle P,\theta,E_i,H_i\rangle \in \bigcup \mathfrak{D}$ are generated by the same element of $\mathcal{A}\}$;
- $H^\vee = (E_S \cup H_S) \setminus E^\vee$ for the corresponding $\langle P,\theta,E_S,H_S\rangle \in \mathcal{A}$.

Intuitively: for each hypothetical answer in $\mathcal{A}$, $\bigvee \mathfrak{D}$ contains the tuple that includes all evidence for $\langle P,\theta\rangle$ that is in *some* element of $\mathfrak{D}$, and $H^\vee$ contains the remaining hypotheses.

Let $\mathcal{D} \in \mathfrak{D}$ and $\langle P,\theta,E,H\rangle \in \mathcal{D}$. Since $\langle P,\theta,E^\vee,H^\vee\rangle \in \bigvee \mathfrak{D}$ and $E \subseteq E^\vee$ by construction, $\mathcal{D} \sqsubseteq \bigvee \mathfrak{D}$. So $\bigvee \mathfrak{D}$ is an upper bound of $\mathfrak{D}$.

Now suppose that $\mathcal{Y} \in \mathfrak{L}$ is an upper bound of $\mathfrak{D}$. We need to show that $\bigvee \mathfrak{D} \sqsubseteq \mathcal{Y}$. For this, choose $\langle P,\theta,E^\vee,H^\vee\rangle \in \bigvee \mathfrak{D}$. For every $\langle P,\theta,E,H\rangle \in \bigcup \mathfrak{D}$ that is generated by the same element of $\mathcal{A}$ as $\langle P, \theta, E^\vee, H^\vee \rangle$, there must exist $\langle P, \theta, E', H' \rangle \in \mathcal{Y}$ such that $E' \cup H' = E \cup H$ and $E \subseteq E'$ (since $\mathcal{Y}$ is an upper bound of $\mathfrak{D}$). Furthermore, this element must be the same for all such tuples, since it is also generated by the same element of $\mathcal{A}$, and $\mathcal{Y}$ has only one element with this property. It follows that $E^\vee = \bigcup E \subseteq E'$. Since this holds for all elements of $\bigvee \mathfrak{D}$, we conclude that $\bigvee \mathfrak{D} \sqsubseteq \mathcal{Y}$, and therefore $\bigvee \mathfrak{D}$ is the least upper bound of $\mathfrak{D}$. □

Corollary 10.

$\langle \mathfrak{L}, \sqsubseteq \rangle$ *is a complete lattice, which we call the* evidence lattice*.*

The negation update operator

The knowledge update operator we consider is defined over a bilattice defined from the evidence lattice. Working with a bilattice allows us to separate reasoning about positive and negative inferred literals.

Definition 11.

The order $\langle \mathbf{S}, \preceq \rangle$ *is defined as follows.*

- $\mathbf{S} \subseteq \mathfrak{L} \times \mathfrak{L}$ *contains all pairs* $\langle \mathcal{X}, \mathcal{Y} \rangle$ *where* $\mathcal{X} \subseteq \mathcal{A}$ *and* $\mathcal{Y} \sqsupseteq \mathcal{A}$.
- $\langle \mathcal{X}, \mathcal{Y} \rangle \preceq \langle \mathcal{X}', \mathcal{Y}' \rangle$ *if* $\mathcal{X} \supseteq \mathcal{X}'$ *and* $\mathcal{Y} \sqsubseteq \mathcal{Y}'$.

Lemma 11.

$\langle \mathbf{S}, \preceq \rangle$ *is a complete lattice.*

Proof. This follows directly from the fact that both projections of $\mathbf{S}$ with the corresponding relations are complete lattices. □

Intuitively, an element $\langle \mathcal{X}, \mathcal{Y} \rangle \in \mathbf{S}$ corresponds to two distinct updates of $\mathcal{A}$: a set of elements $\mathcal{X} \subseteq \mathcal{A}$ that has not been contradicted by any knowledge learned in the update process, and an "updated" version $\mathcal{Y}$ of $\mathcal{A}$ where some hypotheses have been moved to evidence. The negation update operator R, defined below, acts independently in each of these sets, and the operator $\sqcap$, also defined below, combines these updates in a single one.

Definition 12.

The negation update operator $R : \mathbf{S} \to \mathbf{S}$ *is defined as* $R(\mathcal{X}, \mathcal{Y}) = \langle R_1(\mathcal{X}, \mathcal{Y}), R_2(\mathcal{X}, \mathcal{Y}) \rangle$*, where:*

- $R_1(\mathcal{X}, \mathcal{Y})$ *is the result of removing from* $\mathcal{X}$ *the tuples* $\langle P, \theta, E, H \rangle$ *for which there exists an element* $\langle P', \theta', E', \emptyset \rangle \in \mathcal{Y}$ *such that* $\neg P'\theta' \in H$.
- $R_2(\mathcal{X}, \mathcal{Y})$ *is obtained from* $\mathcal{Y}$ *by replacing every tuple* $\langle P, \theta, E, H \rangle$ *with the tuple* $\langle P, \theta, E \cup A, H \setminus A \rangle$ *where* A *is the set of all* $\neg\alpha$ *such that there exists no tuple* $\langle P', \theta', E', H' \rangle \in \mathcal{X}$ *with* $\alpha = P'\theta'$.

Intuitively, R_1 removes tuples in $\mathcal{X}$ that include negative hypotheses disproven in $\mathcal{Y}$, while R_2 updates $\mathcal{Y}$ by moving negative facts to evidence if there is no hypothetical answer for them in $\mathcal{X}$. Keeping these two different updating mechanisms separate simplifies the proof that we can reach a fixpoint.

Definition 13.

For $\langle \mathcal{X}, \mathcal{Y} \rangle \in \mathbf{S}$, define $\mathcal{X} \sqcap \mathcal{Y}$ to be the set of tuples $\langle P, \theta, E, H \rangle \in \mathcal{Y}$ for which there exists $\langle P, \theta, E', H' \rangle \in \mathcal{X}$ with $E \cup H = E' \cup H'$, $E^+ = (E')^+$ and $H^+ = (H')^+$.

Intuitively, if $\langle \mathcal{X}, \mathcal{Y} \rangle \in \mathbf{S}$ corresponds to two distinct updates of $\mathcal{A}$ as described above, then $\mathcal{X} \sqcap \mathcal{Y}$ combines these updates in a generalized hypothetical answer containing the tuples from $\mathcal{A}$ that remain in $\mathcal{X}$ (i.e., that do not contain hypotheses $\neg\alpha$ where α has been proven) with their sets of evidence updated as in $\mathcal{Y}$ (i.e., where hypotheses $\neg\alpha$ where α has been disproven have been moved to evidence).

Lemma 12.

R is monotonic.

Proof. Let $\langle \mathcal{X}, \mathcal{Y} \rangle, \langle \mathcal{X}', \mathcal{Y}' \rangle \in \mathbf{S}$ be such that $\langle \mathcal{X}, \mathcal{Y} \rangle \preceq \langle \mathcal{X}', \mathcal{Y}' \rangle$, i.e., that $\mathcal{X} \supseteq \mathcal{X}'$ and $\mathcal{Y} \sqsubseteq \mathcal{Y}'$. We need to show that $R(\mathcal{X}, \mathcal{Y}) \preceq R(\mathcal{X}', \mathcal{Y}')$, i.e., that $R_1(\mathcal{X}, \mathcal{Y}) \supseteq R_1(\mathcal{X}', \mathcal{Y}')$ and $R_2(\mathcal{X}, \mathcal{Y}) \sqsubseteq R_2(\mathcal{X}', \mathcal{Y}')$.

To show that $R_1(\mathcal{X}, \mathcal{Y}) \supseteq R_1(\mathcal{X}', \mathcal{Y}')$, observe that $R_1(\mathcal{X}, \mathcal{Y}) \subseteq \mathcal{X}$ by definition of R_1. Since $\mathcal{X} \supseteq \mathcal{X}'$, it suffices to show that any tuples that R_1 removes from $\mathcal{X}$ are also removed from $\mathcal{X}'$. Suppose that $\langle P, \theta, E, H \rangle \in \mathcal{X} \setminus R_1(\mathcal{X}, \mathcal{Y})$, i.e. that $\langle P, \theta, E, H \rangle$ is removed from $\mathcal{X}$ by R_1. Then there exists $\langle P', \theta', E', \emptyset \rangle \in \mathcal{Y}$ such that $\neg P'\theta \in H$. Since $\mathcal{Y} \sqsubseteq \mathcal{Y}'$, there must be a tuple $\langle P', \theta', E'', H'' \rangle \in \mathcal{Y}'$ such that $E'' \cup H'' = E' \cup \emptyset = E'$ and $H'' \subseteq \emptyset$. These conditions imply that $H'' = \emptyset$ and $E'' = E'$, i.e. $\langle P', \theta', E', \emptyset \rangle \in \mathcal{Y}'$, and therefore $\langle P, \theta, E, H \rangle$ is also removed from $\mathcal{X}'$ by R_1, i.e. $\langle P, \theta, E, H \rangle \in \mathcal{X}' \setminus R_1(\mathcal{X}', \mathcal{Y}')$. Therefore $R_1(\mathcal{X}, \mathcal{Y}) \supseteq R_1(\mathcal{X}', \mathcal{Y}')$.

To show that $R_2(\mathcal{X}, \mathcal{Y}) \sqsubseteq R_2(\mathcal{X}', \mathcal{Y}')$, choose $\langle P, \theta, E, H \rangle \in R_2(\mathcal{X}, \mathcal{Y})$. Then there exists a tuple $\langle P, \theta, E_Y, H_Y \rangle \in \mathcal{Y}$ such that $E = E_Y \cup A$ and $H = H_Y \setminus A$ for some set of literals A. Furthermore, if $\neg\alpha \in A$, then there exists no tuple $\langle P^*, \theta^*, E^*, H^* \rangle \in \mathcal{X}$ with $\alpha = P^*\theta^*$. Since $\mathcal{X} \subseteq \mathcal{X}'$, there is no such tuple in $\mathcal{X}'$, either. Also, since $\mathcal{Y} \sqsubseteq \mathcal{Y}'$, there exists $\langle P, \theta, E', H' \rangle \in \mathcal{Y}'$ such that $E_Y \cup H_Y = E' \cup H'$ and $E_Y \subseteq E'$. Therefore, $R_2(\mathcal{X}', \mathcal{Y}')$ contains $\langle P, \theta, E' \cup A', H \setminus A' \rangle$ for some $A' \supseteq A$. But $(E' \cup A') \cup (H' \setminus A') = E' \cup H' = E_Y \cup H_Y = (E_Y \cup A) \cup (H_Y \setminus A) = E \cup H$ and $E = E_Y \cup A \subseteq E' \cup A'$, therefore $R_2(\mathcal{X}, \mathcal{Y}) \sqsubseteq R_2(\mathcal{X}', \mathcal{Y}')$.

Therefore R is a monotonic operator. $\square$

Corollary 13.

The negation update operator R has a least fixpoint.

Proof. Consequence of the previous lemma and the Knaster–Tarski theorem. $\square$

Definition 14.

The set $\mathcal{S}^{\downarrow}_{\tau+1}$ is defined as $\mathcal{X}_0 \sqcap \mathcal{Y}_0$, where $\langle \mathcal{X}_0, \mathcal{Y}_0 \rangle$ is the least fixpoint of the negation update operator R.

For convenience, we write simply $\mathcal{S}^{\downarrow}$ for $\mathcal{S}^{\downarrow}_{\tau+1}$ in the remainder of this section. Recall that the set $\mathcal{A}$ was constructed from $\mathcal{S}^{\downarrow}_{\tau}$ by updating its tuples with the (positive) evidence produced by the datastream at time point τ. The set $\mathcal{S}^{\downarrow}$ corresponds to updating $\mathcal{A}$ with all the information about negative hypotheses that can be proved or disproved (by iterating R and combining the result using $\sqcap$).

In particular, (i) $\mathcal{S}^{\downarrow} \subseteq \mathcal{Y}_0$ and (ii) $\mathcal{X}_0 \sqsubseteq \mathcal{S}^{\downarrow}$. (Note however that $\mathcal{S}^{\downarrow} \notin \mathbf{S}$.)

Lemma 14.

$\langle \mathcal{S}^{\downarrow}, \mathcal{S}^{\downarrow} \rangle$ is a fixpoint of R.

Proof. We need to show that $\langle \mathcal{S}^{\downarrow}, \mathcal{S}^{\downarrow} \rangle = R(\mathcal{S}^{\downarrow}, \mathcal{S}^{\downarrow})$, i.e. that $\mathcal{S}^{\downarrow} = R_i(\mathcal{S}^{\downarrow}, \mathcal{S}^{\downarrow})$ for $i = 1, 2$. Let $\langle \mathcal{X}_0, \mathcal{Y}_0 \rangle$ be the least fixpoint of R, as in the definition of $\mathcal{S}^{\downarrow}$.

$R_1(\mathcal{S}^{\downarrow}, \mathcal{S}^{\downarrow})$ is obtained by removing from $\mathcal{S}^{\downarrow}$ the tuples $\langle P, \theta, E, H \rangle$ for which there exists $\langle P', \theta', E', \emptyset \rangle \in \mathcal{S}^{\downarrow}$ such that $\neg P'\theta \in H$. But such a tuple $\langle P', \theta', E', \emptyset \rangle$ would also be in $\mathcal{Y}_0$ by (i), and by (ii) this would imply that $R_1(\mathcal{X}_0, \mathcal{Y}_0) \neq \mathcal{X}_0$ (because it would lead to some tuple in $\mathcal{X}_0$ being removed by R_1), which contradicts $\langle \mathcal{X}_0, \mathcal{Y}_0 \rangle$ being a fixpoint of R.

$R_2(\mathcal{S}^{\downarrow}, \mathcal{S}^{\downarrow})$ is obtained by updating each tuple $\langle P, \theta, E, H \rangle \in \mathcal{S}^{\downarrow}$ by moving literals of the form $\neg\alpha \in H$ to E if there exists no tuple $\langle P', \theta', E', H' \rangle \in \mathcal{S}^{\downarrow}$ with $\alpha = P'\theta'$. By (ii) this implies that no such tuple exists in $\mathcal{X}_0$ either (note that the condition does not impose restrictions on E' and H', so it does not matter that these sets may differ in the actual element of $\mathcal{X}_0$). By (i) any tuple updated in the computation of $R_2(\mathcal{S}^{\downarrow}, \mathcal{S}^{\downarrow})$ would therefore also be updated when computing $R_2(\mathcal{X}_0, \mathcal{Y}_0)$, which again would contradict $\langle \mathcal{X}_0, \mathcal{Y}_0 \rangle$ being a fixpoint of R. □

It can also be shown that R is continuous, and therefore its least fixpoint is equal to $R^{\omega}(\mathcal{A}, \mathcal{A})$ in the transfinite sequence of iterates of R starting from $(\mathcal{A}, \mathcal{A})$. However, this is immaterial for our presentation, and we skip the formal proof.

Soundness and completeness

We now show that $\mathcal{S}^{\downarrow}_{\tau}$ is the largest generalized hypothetical answer to Q over D_τ for which the converse of Proposition 5 holds, i.e. such that: if $\langle P, \theta, E, H \rangle \in \mathcal{S}^{\downarrow}_{\tau}$, then there exists an SLDNF-refutation with future premises for $\langle P, \Pi \rangle$ over D_τ with answer θ. Observe that this holds for $\mathcal{S}^{\downarrow}_{-1}$ by construction, using soundness and completeness of $\mathcal{P}_Q$ (see [20]).

Lemma 15.

If $\langle P, \theta, E, H \rangle \in \mathcal{S}^{\downarrow}_{\tau}$ or $\langle P, \theta, E, H \rangle \in \mathcal{A}_{\tau}$, then $E^{+} \subseteq D_\tau$ and every element of H^{+} is future-possible w.r.t. τ.

Proof. Straightforward by induction on τ, using the definition of $\mathcal{S}^{\downarrow}_{\tau}$ and $\mathcal{A}_{\tau}$. □

Next, we show that the following property also holds, by induction on τ:

$$\boxed{\begin{minipage}{0.75\textwidth}Let D' be a dataset such that $D' \Supset_\tau D$. Then $\langle P, \theta, E, H\rangle \in \mathcal{S}^\downarrow_\tau$ with $H^+ \subseteq (D' \setminus D_\tau)$ iff there exists a derivation $\mathcal{D}$ such that (i) $\mathcal{D}$ is an SLDNF-derivation proving that $\Pi \cup D' \models P\theta$ and (ii) $\mathcal{D}$ is an SLD$^\neg$-derivation proving that $\Pi \cup E \cup H \models P\theta$ that uses all elements of $E \cup H$.\end{minipage}} \qquad (*)$$

In this property, an SLD$^\neg$-derivation is an SLD-derivation where negated atoms are treated by checking whether they appear as facts (in either E^- or H^-). Derivation $\mathcal{D}$ "uses" a (negated) fact if that fact is unified in at least one step of $\mathcal{D}$.

For $\mathcal{S}^\downarrow_{-1}$, this property is again a straightforward consequence of how pre-processing is defined; the interested reader can find a proof in our previous work [20].

The induction step proceeds in two parts. First, we show that the construction of the auxiliary set $\mathcal{A}_{\tau+1}$, obtained by updating the positive part of $\mathcal{S}^\downarrow_\tau$ with the information in $D|_{\tau+1}$, preserves property $(*)$.

Lemma 16.

If $\mathcal{S}^\downarrow_\tau$ satisfies $(*)$, then $\mathcal{A}_{\tau+1}$ satisfies $(*)$.

Proof. Let D' be a dataset such that $D' \Supset_{\tau+1} D$. Then also $D' \Supset_\tau D$.

To prove the direct implication in $(*)$, assume that $\langle P, \theta, E, H\rangle \in \mathcal{A}_{\tau+1}$ is such that $H^+ \subseteq D' \setminus D_{\tau+1}$. By construction of $\mathcal{A}_{\tau+1}$, there exists $\langle P, \theta, E', H'\rangle \in \mathcal{S}^\downarrow_\tau$ such that $E = E' \cup ({H'}^+ \cap D|_{\tau+1})$ and $H = H' \setminus D|_{\tau+1}$. Any elements in $H' \setminus H$ must be in $D|_{\tau+1}$, so $H' \subseteq D' \setminus D_\tau$.

By hypothesis on $\mathcal{S}^\downarrow_\tau$, there exists a derivation $\mathcal{D}$ such that $\mathcal{D}$ is an SLDNF-derivation showing that $\Pi \cup D' \models P\theta$ and $\mathcal{D}$ is an SLD$^\neg$-derivation proving that $\Pi \cup E' \cup H' \models P\theta$ that uses all elements of $E' \cup H'$. But $E' \cup H' = E \cup H$, so $\mathcal{D}$ is also an SLD$^\neg$-derivation proving that $\Pi \cup E \cup H \models P\theta$ that uses all elements of $E \cup H$.

For the converse implication, let $\mathcal{D}$ be an SLDNF-derivation proving that $\Pi \cup D' \models P\theta$, and let F contain the set of elements of D' that are used in $\mathcal{D}$ and all negative literals that appear in $\mathcal{D}$. Then $\mathcal{D}$ is also an SLD$^\neg$-derivation proving that $\Pi \cup F \models P\theta$. By hypothesis on $\mathcal{S}^\downarrow_\tau$, there exists a tuple $\langle P, \theta, E, H\rangle \in \mathcal{S}^\downarrow_\tau$ such that $H^+ \subseteq D' \setminus D_\tau$ and $E \cup H = F$.

If $\langle P, \theta, E, H\rangle \notin \mathcal{A}_{\tau+1}$, then $H^+ \setminus D_{\tau+1}$ contains some elements that are not future-possible w.r.t. $\tau+1$; such elements cannot be in D', which is a contradiction (since they are in F, they are used in $\mathcal{D}$, but they cannot be unified with any element of $\Pi \cup D'$).

Therefore $\mathcal{A}_{\tau+1}$ also contains an element $\langle P, \theta, E', H'\rangle$ with $E' = E \cup (H^+ \cap D|_{\tau+1})$, $H' = H \setminus D|_{\tau+1}$, and such that $H' \subseteq D' \setminus D_{\tau+1}$. As a consequence, $E \cup H = E' \cup H' = F$, establishing the thesis. □

Next, we show that $(*)$ is preserved in the construction of $\mathcal{S}^\downarrow_{\tau+1}$ from $\mathcal{A}_{\tau+1}$.

The proof uses transfinite induction. We again split it in several lemmas for simplicity. The base case is trivial, since the least element of $\mathbf{S}$ is $\langle \mathcal{A}_{\tau+1}, \mathcal{A}_{\tau+1}\rangle$ and trivially $\mathcal{A}_{\tau+1} \sqcap \mathcal{A}_{\tau+1} = \mathcal{A}_{\tau+1}$.

Lemma 17.

Let $\langle \mathcal{X}, \mathcal{Y}\rangle \in \mathbf{S}$ be such that: (i) if $\langle P, \theta, E, H\rangle \in \mathcal{X} \sqcap \mathcal{Y}$, then $E^+ \subseteq D_{\tau+1}$ and (ii) $\mathcal{X} \sqcap \mathcal{Y}$ satisfies $(*)$. Then $R_1(\mathcal{X}, \mathcal{Y}) \sqcap R_2(\mathcal{X}, \mathcal{Y})$ satisfies $(*)$.

Proof. Let D' be a datastream such that $D' \Supset_{\tau+1} D$.

For the direct implication in $(*)$, choose $\langle P, \theta, E, H\rangle \in R_1(\mathcal{X}, \mathcal{Y}) \sqcap R_2(\mathcal{X}, \mathcal{Y})$. In particular $\langle P, \theta, E, H\rangle \in R_2(\mathcal{X}, \mathcal{Y})$, so there exists a tuple $\langle P, \theta, E', H'\rangle \in \mathcal{X} \sqcap \mathcal{Y}$ such that $E \cup H = E' \cup H'$ and $(H')^+ = H^+$. The hypothesis on $\langle \mathcal{X}, \mathcal{Y}\rangle$ immediately establishes the thesis.

For the converse implication, assume that $\mathcal{D}$ is an SLDNF-derivation showing that $\Pi \cup D' \models P\theta$ and define F as in the proof of Lemma 16, i.e. as the set of elements of D' that are used in $\mathcal{D}$ together with all negative literals that appear in $\mathcal{D}$.

By hypothesis there exists a tuple $\langle P, \theta, E, H\rangle \in \mathcal{X} \sqcap \mathcal{Y}$ such that $H^+ \subseteq (D' \setminus D_{\tau+1})$ and $E \cup H = F$. By definition of $\sqcap$, there exists a tuple $\langle P, \theta, E_X, H_X\rangle \in \mathcal{X}$ such that $E_X \cup H_X = E \cup H = F$, and $\langle P, \theta, E, H\rangle \in \mathcal{Y}$. By definition of R_2, there is also a tuple $\langle P, \theta, E_Y, H_Y\rangle \in R_2(\mathcal{X}, \mathcal{Y})$ such that $E_Y \cup H_Y = E \cup H = F$ and $H_Y^+ = H^+$. To establish the thesis, all that is left to show is that $\langle P, \theta, E_X, H_X\rangle \in R_1(\mathcal{X}, \mathcal{Y})$.

Assume towards a contradiction that this is not the case. Then there exists an element $\langle P', \theta', E', \emptyset\rangle \in \mathcal{Y}$ with $\neg P'\theta' \in H_X$. Since $\emptyset^+ \subseteq D' \setminus D_{\tau+1}$ and $\langle \mathcal{X}, \mathcal{Y}\rangle$ satisfies $(*)$, there exists an SLDNF-derivation $\mathcal{D}'$ showing that $\Pi \cup D' \models P'\theta'$, which contradicts the fact that $\mathcal{D}$ at some point must process the fact $\neg P'\theta'$ by showing that no such derivation exists. □

Lemma 18.

Let $\{\langle \mathcal{X}_i, \mathcal{Y}_i\rangle \mid i \in I\} \subseteq \mathbf{S}$ be such that $\mathcal{X}_i \sqcap \mathcal{Y}_i$ satisfies $(*)$ for all $i \in I$, and define $\langle \mathcal{X}, \mathcal{Y}\rangle = \bigvee\{\langle \mathcal{X}_i, \mathcal{Y}_i\rangle \mid i \in I\}$. Then $\mathcal{X} \sqcap \mathcal{Y}$ satisfies $(*)$.

Proof. Let D' be a datastream, and assume that $D' \Supset_{\tau+1} D$.

For the direct implication in $(*)$, pick $\langle P, \theta, E, H\rangle \in \mathcal{X} \sqcap \mathcal{Y}$ with $H^+ \subseteq D' \setminus D_{\tau+1}$. Then $\langle P, \theta, E, H\rangle \in \mathcal{Y}_i$ for all i, and every $\mathcal{X}_i$ contains an element $\langle P, \theta, E_i, H_i\rangle$ with $E_i \cup H_i = E \cup H$ and $E_i \subseteq E$. Applying the hypothesis for any i immediately establishes the thesis.

Conversely, let $\mathcal{D}$ be an SLDNF-derivation establishing $\Pi \cup D' \models P\theta$ and define F from $\mathcal{D}$ again as in the proofs of Lemmas 16 and 17. By hypothesis, for each i there must exist $\langle P, \theta, E_i, H_i\rangle \in \mathcal{X}_i \sqcap \mathcal{Y}_i$ with $E_i \cup H_i = F$ and such that $H_i^+ \subseteq D' \setminus D_{\tau+1}$.

This means that $\langle P, \theta, E_i, H_i\rangle \in \mathcal{Y}_i$ for each i, and there exists $\langle P, \theta, E_X, H_X\rangle$ such that, for every i, $E_X \cup H_X = E_i \cup H_i$ and $\langle P, \theta, E_X, H_X\rangle \in \mathcal{X}_i$. Then $\langle P, \theta, E_X, H_X\rangle \in \mathcal{X}$, and there is a tuple $\langle P, \theta, E_Y, H_Y\rangle \in \bigvee \mathcal{Y}_i$ such that $\langle P, \theta, E_i, H_i\rangle \sqsupseteq \langle P, \theta, E_Y, H_Y\rangle$. In particular, $E_Y \cup H_Y = E_i \cup H_i$ for all i, and therefore also $E_Y \cup H_Y = E_X \cup H_X$. Thus $\langle P, \theta, E_Y, H_Y\rangle \in \mathcal{X} \sqcap \mathcal{Y}$, and since $E_X \cup H_X = F$ and $H_X^+ = H_i^+ \subseteq D' \setminus D_{\tau+1}$ (for an arbitrary i) we can conclude that the thesis holds. □

Corollary 19.

If $\mathcal{A}_{\tau+1}$ satisfies $(*)$, then $\mathcal{S}^{\downarrow}_{\tau+1}$ also satisfies $(*)$.

A simple consequence of these results is the following theorem.

Theorem 3.

For every τ, the set $\mathcal{S}_\tau^\downarrow$ coincides with the set $\mathcal{S}_\tau^{\mathrm{SLDNF}}$ (Definition 8).

Proof. By Corollary 19, $\mathcal{S}_\tau^\downarrow$ satisfies property $(*)$. In particular, if $\langle P_0, \theta, E, H\rangle \in \mathcal{S}_\tau^\downarrow$ then there exists an SLDNF-derivation proving that $\Pi \cup D \cup H^+ \models P\theta$, which can readily be made into an SLDNF-derivation with future premises for Q_0 over D_τ by removing any unification steps with elements of H^+. Therefore $\mathcal{S}_\tau^\downarrow \subseteq \mathcal{S}_\tau^{\mathrm{SLDNF}}$.

Conversely, assume that $\langle P, \theta, E, H\rangle \in \mathcal{S}_\tau^{\mathrm{SLDNF}}$. Then there exists an SLDNF-refutation with future premises for Q_0 over D_τ containing a subderivation starting from goal $\neg P$ with a leaf computing the answer θ, containing exactly the literals in H, and such that E contains all literals that were removed either by unification with an element of D_τ or through a failed auxiliary derivation. Extending this branch with unification steps with all elements of H yields an SLD$^\neg$-derivation proving that $\Pi \cup E \cup H \models P\theta$ that uses all elements of $E \cup H$; furthermore, this is also an SLDNF-derivation proving that $\Pi \cup D_\tau \cup H \models P\theta$. Therefore the two conditions in property $(*)$ are satisfied, which implies that $\langle P, \theta, E, H\rangle \in \mathcal{S}_\tau^\downarrow$. □

Corollary 20.

$\mathcal{S}_\tau^\downarrow$ is a generalized hypothetical answer to Q over D_τ.

Proof. From Theorem 3 and Proposition 5. □

6 Incrementally computing $\mathcal{S}_\tau^\downarrow$

The sequence $\mathcal{S}_\tau^\downarrow$ defined in the previous section represents all possible SLDNF-refutations with future premises for the query of interest at each time point, given the dataset D. However, $\mathcal{S}_\tau^\downarrow$ is usually an infinite set, which makes it inconvenient in practice. Furthermore, it contains irrelevant information, as we are primarily interested in supported answers.

In this section we introduce an algorithm that computes the "interesting" part of $\mathcal{S}_\tau^\downarrow$ symbolically, by using variables to represent potentially infinite sets of hypothetical answers. This computation is incremental, with the set being updated at each time point with the new information arriving at the data stream. We continue using the notations introduced in the previous section.

Definition 15.

A schematic hypothetical answer *to a set of queries Q over D_τ is a set S of tuples $\langle P', \theta, E, H\rangle$ such that the set S' of all ground instances of all elements of S is a generalized hypothetical answer to Q.*

We also say that S represents *the generalized hypothetical answer S'.*

Example 11.

Let $Q = \langle P, \Pi\rangle$ be a query and $\mathsf{Q} = \{Q_i \mid i \in I\}$ with $Q_i = \langle P_i, \Pi\rangle$ be the set of *queries generated by pre-processing Q. The set of all tuples $\langle P_i, \theta, \emptyset, H\rangle$ such that $\langle \theta, H\rangle \in \mathcal{P}_{Q_i}$ for all $i \in I$ is a schematic hypothetical answer to* Q *that represents $\mathcal{S}^{\downarrow}_{-1}$ (Definition 9).* $\triangleleft$

Local most general unifiers

Before we introduce our algorithmic construction, we need an auxiliary definition.

Definition 16.

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 12 below.

Lemma 21.

Let Γ and Δ be finite sets of atoms such that all atoms in Δ are ground. 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 $\Psi = \{a \in \Gamma \mid a\sigma \in \Delta\}$. 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. $\square$

Example 12.

Consider the program Π consisting of the rule

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

where p *is an EDB with* $\delta(p) = 2$ *and* r *is an EDB with* $\delta(r) = 0$*, and the query* $Q = \langle q(X,T), \Pi\rangle$*. 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* $D|_0$*:*

$$\begin{tikzcd}
\leftarrow p(X,T), r(Y,T) \arrow[d, "{[X{:=}\mathsf{a},T{:=}0]}"] & \leftarrow p(X,T), r(Y,T) \arrow[d, "{[Y{:=}\mathsf{b},T{:=}0]}"] \\
\leftarrow r(Y,0) \arrow[d, "{[Y{:=}\mathsf{b}]}"] & \leftarrow p(X,0) \arrow[d, "{[X{:=}\mathsf{a}]}"] \\
\Box & \Box
\end{tikzcd}$$

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.* $\triangleleft$

The algorithm

We now combine local mgus to deal with communication delays in the positive fragment of our language, with our original proposal for dealing with negation [20]. The trick is to balance the amount of information in the schematic hypothetical answers computed online, so that the updating procedure terminates but negations are updated correctly.

We achieve this by: (i) ensuring that we introduce schematic hypothetical answers for every query that may need to be examined when updating negations, even if there is no evidence for them, and (ii) restricting the negated hypotheses that are updated to those whose timestamp is at most the current one.

The algorithm uses the notion of potentially future atoms, which generalize future-possible atoms to non-ground atoms.

Definition 17.

An atom $p(t_1,\ldots,t_n)$ *is a* potentially future atom *w.r.t. time point* τ *if* $\tau < t_n + \delta(p(t_1,\ldots,t_n))$ *or* t_n *contains a variable.*

A potentially future atom is future-possible for at least some instantiation of its variables.

Algorithm 1.

We initialize $\mathcal{S}_{-1} = \emptyset$*.*

1. Define a set $\mathcal{B}_\tau$ updating $\mathcal{S}_{\tau-1}$ with information from the datastream.
   (a) *For each $Q \in \mathsf{Q}$, if $\langle\theta, H\rangle \in \mathcal{P}_Q$ and σ is a local mgu for H and $D|_\tau$ such that $H \cap D|_\tau \neq \emptyset$ and $H^+\sigma \setminus D|_\tau$ only contains potentially future atoms w.r.t. τ, then $\langle P, \theta\sigma, H\sigma \cap D|_\tau, H\sigma \setminus D|_\tau\rangle \in \mathcal{B}_\tau$, where $Q = \langle P, \Pi\rangle$.*
   (b) *If $\langle P, \theta, E, H\rangle \in \mathcal{S}_{\tau-1}$ and σ is a local mgu for H and $D|_\tau$ such that the set $H^+\sigma \setminus D|_\tau$ only contains potentially future atoms w.r.t. τ, then $\langle P, \theta\sigma, E \cup E', H\sigma \setminus D|_\tau\rangle \in \mathcal{B}_\tau$, where $E' = H\sigma \cap D|_\tau$.*
2. Add answers to queries that might be examined when updating negations.
   For each $Q \in \mathsf{Q}$ with $Q = \langle P, \Pi\rangle$, let $\sigma = [T := \tau]$ with T the temporal variable in P. If $\langle\theta, H\rangle \in \mathcal{P}_Q$ and every element of $H\sigma$ is either potentially future w.r.t. τ or negated, then add $\langle P, \theta\sigma, \emptyset, H\sigma\rangle$ to $\mathcal{B}_\tau$.
3. Process negated literals.
   Fix a topological ordering of the stratification of $\Pi^\downarrow$. There is only a finite number of predicate symbols p_t such that $\mathcal{B}_\tau$ contains at least one tuple $\langle P, \theta, E, H\rangle$ where P is an atom built from p and $T\theta = t$ (where T is the temporal parameter in P). For each of these p_t in order:
   (a) *define L to be the set of elements $\langle P, \theta, E, H\rangle$ in $\mathcal{B}_\tau$ such that $T\theta = t$;*
   (b) *if L contains a tuple $\langle P, \theta, E', \emptyset\rangle$, then, in $\mathcal{B}_\tau$, replace every $\langle P', \sigma, E, H\rangle$ such that $P\sigma$ is unifiable with an element $h \in H^-$ with all possible $\langle P', \sigma\theta', E\theta', H\theta'\rangle$ such that θ' is a minimal substitution with the property that $P\theta$ is not unifiable with $h\theta'$;*
   (c) *for each $\langle P, \theta, E, H\rangle \in \mathcal{B}_\tau$, if H^- contains an element h with predicate symbol p and timestamp $t \leq \tau$ and there is no tuple $\langle P, \theta', E', H'\rangle \in L$ such that h and $P\theta'$ are unifiable, then remove $\neg h$ from H and add it to E.*

   The resulting set, after all predicate symbols have been processed, is $\mathcal{S}_\tau$.

Step (1) is essentially the original update step [17], which compares the facts delivered by the data stream with the positive hypotheses in the current schematic hypothetical answers. The hypotheses that have arrived are moved to evidence, while those answers containing hypotheses that no longer have a chance of being delivered are discarded. The use of local mgus ensures that no answers are lost even in presence of communication delays.

Step (2), adapted from the original algorithm [20], guarantees that schematic hypothetical answers are added for any queries that may be evaluated when updating negated hypotheses – so their absence guarantees that they have been removed in a previous iteration. Step (3), also adapted from the original algorithm [20], updates negated hypotheses.

For positive programs, the set $\mathcal{S}_\tau$ corresponds precisely (modulo instantiation) to the subset of elements $\langle P, \theta, E, H\rangle \in \mathcal{S}^\downarrow_\tau$ where either $E \neq \emptyset$ or $T\theta \leq \tau$ (Proposition 22 below). For programs using negation, the correspondence is more subtle, and we discuss it after illustrating the algorithm with our running example.

Example 13.

We illustrate this mechanism again using the program from Example 1. Recall that there are two queries: the main query $Q_{OK} = \langle \mathsf{OK}(X, T), \Pi_{WT}\rangle$ and the auxiliary *query* $Q_S = \langle \mathsf{Shdn}(X,T), \Pi_{WT}\rangle$. *Pre-processing these queries (see Example 4) yields*

$$\begin{aligned} P_{Q_{OK}} &= \{\langle \emptyset, \{\neg\mathsf{Shdn}(X,T+1)\}\rangle\} \\ P_{Q_S} &= \{\langle \emptyset, \{\mathsf{Hot}(X,T-2), \mathsf{Hot}(X,T-1), \mathsf{Hot}(X,T)\}\rangle\} \end{aligned}$$

We consider three turbines $\mathsf{wt}i$ *with* $i = 1,2,3$, *with different associated communication delays:* $\delta(\mathsf{Hot}(\mathsf{wt1},T)) = 2$, $\delta(\mathsf{Hot}(\mathsf{wt2},T)) = 3$, *and* $\delta(\mathsf{Hot}(\mathsf{wt3},T)) = 1$. *The facts delivered by the datastream are:*

$$\begin{aligned} D|_0 &= \{\mathsf{Hot}(\mathsf{wt1},0)\} & D|_2 &= \{\mathsf{Hot}(\mathsf{wt1},2), \mathsf{Hot}(\mathsf{wt2},0)\} \\ D|_1 &= \emptyset & D|_3 &= \{\mathsf{Hot}(\mathsf{wt1},1), \mathsf{Hot}(\mathsf{wt3},2)\} \\ & & D|_4 &= \emptyset \end{aligned}$$

- *At* $\tau = 0$: $D|_0 = \{\mathsf{Hot}(\mathsf{wt1},0)\}$. *Step 1(a) starts by unifying* $D|_0$ *with the different sets of hypotheses in* $P_{Q_{OK}}$ and P_{Q_S}. *The only case where there is a successful step of SLD-resolution is for* P_{Q_S}, *yielding the local mgu* $[X := \mathsf{wt1}, T := 2]$. *Therefore, after this step the set* $\mathcal{B}_0$ *contains*

  $$\langle \mathsf{Shdn}(X,T), [X := \mathsf{wt1}, T := 2], \{\mathsf{Hot}(\mathsf{wt1},0)\}, \{\mathsf{Hot}(\mathsf{wt1},1), \mathsf{Hot}(\mathsf{wt1},2)\}\rangle .$$

  Step 1(b) is not applicable, since $\mathcal{S}_{-1} = \emptyset$.
  In step 2, we consider the substitution $\theta_0 = [T := 0]$. *The corresponding partial instantiation of the hypothetical answer in* $P_{Q_{OK}}$ *only has negated atoms, so* $\mathcal{B}_0$ *is extended with*

  $$\langle \mathsf{OK}(X,T), [T := 0], \emptyset, \{\neg\mathsf{Shdn}(X,1)\}\rangle .$$

  Applying θ_0 *to the only element of* P_{Q_S} *yields terms with invalid timestamps* (-1 *and* -2), *so we ignore them.*
  For step 3 we fix the trivial topological ordering OK_0. *None of the substeps apply, so we obtain*

  $$\begin{aligned} \mathcal{S}_0 = \{&\langle \mathsf{Shdn}(X,T), [X := \mathsf{wt1}, T := 2], \{\mathsf{Hot}(\mathsf{wt1},0)\}, \{\mathsf{Hot}(\mathsf{wt1},1), \mathsf{Hot}(\mathsf{wt1},2)\}\rangle, \\ &\langle \mathsf{OK}(X,T), [T := 0], \emptyset, \{\neg\mathsf{Shdn}(X,1)\}\rangle\} . \end{aligned}$$

- *At* $\tau = 1$: $D|_1 = \emptyset$. *Thus, step 1(a) is never applicable. All positive hypotheses in the elements of* $\mathcal{S}_0$ *are potentially future atoms w.r.t. 1, therefore we set* $\mathcal{B}_1 = \mathcal{S}_0$. *The application of step 2 is similar to the previous one, extending* $\mathcal{B}_1$ *with*

  $$\langle \mathsf{OK}(X,T), [T := 1], \emptyset, \{\neg\mathsf{Shdn}(X,2)\}\rangle .$$

  For step 3, we consider the topological ordering $\mathsf{OK}_0 < \mathsf{OK}_1$. *The schematic hypothetical answer for* OK_0 *includes the negated hypothesis* $\neg\mathsf{Shdn}(X,1)$, *for which there are no schematic hypothetical answers in* $\mathcal{B}_1$. *Therefore this hypothesis is moved to evidence (step 3 (c)). The schematic hypothetical answer for* OK_1 *includes a negated hypothesis with timestamp higher than* $\tau = 1$, *so it is not changed.*

At the end of this iteration, we obtain

$$\begin{aligned}\mathcal{S}_1 = \{&\langle \mathsf{Shdn}(X,T), [X := \mathsf{wt1}, T := 2], \{\mathsf{Hot}(\mathsf{wt1},0)\}, \{\mathsf{Hot}(\mathsf{wt1},1), \mathsf{Hot}(\mathsf{wt1},2)\}\rangle,\\ &\langle \mathsf{OK}(X,T), [T := 0], \{\neg\mathsf{Shdn}(X,1)\}, \emptyset\rangle,\\ &\langle \mathsf{OK}(X,T), [T := 1], \emptyset, \{\neg\mathsf{Shdn}(X,2)\}\rangle\}\,.\end{aligned}$$

At this point, the system can output the answer $[T := 0]$ *to the original query* Q_{OK}*, reflecting the fact that all turbines are working normally at time point* 0 *(there are no shutdowns at time point* 1*).*

- *At* $\tau = 2$*:* $D|_2 = \{\mathsf{Hot}(\mathsf{wt1},2), \mathsf{Hot}(\mathsf{wt2},0)\}$*. Step 1(a) now yields several different tuples. The fact* $\mathsf{Hot}(\mathsf{wt1},2)$ *can unify with any of the three elements in the hypothetical answer in* P_{Q_S}*, yielding three potential new elements for* $\mathcal{B}_2$*:*

$$\begin{aligned}&\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt1}, T := 2], \{\mathsf{Hot}(\mathsf{wt1},2)\}, \{\mathsf{Hot}(\mathsf{wt1},0), \mathsf{Hot}(\mathsf{wt1},1)\}\rangle\\ &\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt1}, T := 3], \{\mathsf{Hot}(\mathsf{wt1},2)\}, \{\mathsf{Hot}(\mathsf{wt1},1), \mathsf{Hot}(\mathsf{wt1},3)\}\rangle\\ &\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt1}, T := 4], \{\mathsf{Hot}(\mathsf{wt1},2)\}, \{\mathsf{Hot}(\mathsf{wt1},3), \mathsf{Hot}(\mathsf{wt1},4)\}\rangle\end{aligned}$$

The first tuple, however, is discarded because $\delta(\mathsf{Hot}(\mathsf{wt1},0)) = 2$*, so this atom is not potentially future w.r.t. 2. Unification of* $\mathsf{Hot}(\mathsf{wt2},0)$ *with the element of* P_{Q_S}*, on the other hand, yields the tuple*

$$\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt2}, T := 2], \{\mathsf{Hot}(\mathsf{wt2},0)\}, \{\mathsf{Hot}(\mathsf{wt2},1), \mathsf{Hot}(\mathsf{wt2},2)\}\rangle\,.$$

Step 1(b) moves the hypothesis $\mathsf{Hot}(\mathsf{wt1},2)$ *to evidence in the schematic hypothetical answer where it appears, copying the remaining elements of* $\mathcal{S}_1$ *to* $\mathcal{B}_2$ *unchanged. Step 2 is similar to the previous cases, except that now we also generate an answer from* P_{Q_S} *– note that* $\delta(\mathsf{Hot}(X,0)) = 3$*. At this stage, we have*

$$\begin{aligned}\mathcal{B}_2 = \{&\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt1}, T := 3], \{\mathsf{Hot}(\mathsf{wt1},2)\}, \{\mathsf{Hot}(\mathsf{wt1},1), \mathsf{Hot}(\mathsf{wt1},3)\}\rangle,\\ &\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt1}, T := 4], \{\mathsf{Hot}(\mathsf{wt1},2)\}, \{\mathsf{Hot}(\mathsf{wt1},3), \mathsf{Hot}(\mathsf{wt1},4)\}\rangle,\\ &\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt2}, T := 2], \{\mathsf{Hot}(\mathsf{wt2},0)\}, \{\mathsf{Hot}(\mathsf{wt2},1), \mathsf{Hot}(\mathsf{wt2},2)\}\rangle,\\ &\langle \mathsf{Shdn}(X,T), [X := \mathsf{wt1}, T := 2], \{\mathsf{Hot}(\mathsf{wt1},0), \mathsf{Hot}(\mathsf{wt1},2)\}, \{\mathsf{Hot}(\mathsf{wt1},1)\}\rangle,\\ &\langle \mathsf{OK}(X,T), [T := 0], \{\neg\mathsf{Shdn}(X,1)\}, \emptyset\rangle,\\ &\langle \mathsf{OK}(X,T), [T := 1], \emptyset, \{\neg\mathsf{Shdn}(X,2)\}\rangle,\\ &\langle \mathsf{OK}(X,T), [T := 2], \emptyset, \{\neg\mathsf{Shdn}(X,3)\}\rangle,\\ &\langle \mathsf{Shdn}(X,T), [T := 2], \emptyset, \{\mathsf{Hot}(X,0), \mathsf{Hot}(X,1), \mathsf{Hot}(X,2)\}\rangle\}\,.\end{aligned}$$

For step 3, the topological ordering we choose must respect the constraint $\mathsf{Shdn}_2 < \mathsf{OK}_1$*. We use* $\mathsf{Shdn}_2 < \mathsf{OK}_0 < \mathsf{OK}_1 < \mathsf{OK}_2$*. For* Shdn_2 *and* OK_0 *no changes are made: the schematic hypothetical answers associated with the former all have a non-empty set of positive hypotheses, while the latter does not appear as a negated hypotheses in any other schematic hypothetical answer.*

For OK_1*, we need to look at the schematic hypothetical answers that unify with* $\mathsf{Shdn}(X, 2)$*. The last element in* $\mathcal{B}_2$ *still allows for the possibility that any instance of this fact may be proved at a later stage, so we do not make any changes. For* OK_2*, the algorithm also makes no changes as the negative hypothesis in its schematic hypothetical answer has a timestamp higher than* 2*.*
Therefore $\mathcal{S}_2 = \mathcal{B}_2$*.*

- *At* $\tau = 3$*:* $D|_3 = \{\mathsf{Hot}(\mathsf{wt1}, 1), \mathsf{Hot}(\mathsf{wt3}, 2)\}$*. We summarize the result of steps 1 and 2, since the reasoning is very similar to the previous cases. For step 1(a), note that* $\mathsf{Hot}(\mathsf{wt1}, 1)$ *generates no tuples (since all possibilities include atoms that are not potentially future w.r.t. 3); likewise,* $\mathsf{Hot}(\mathsf{wt3}, 2)$ *only generates one tuple, since* $\delta(\mathsf{Hot}(\mathsf{wt3}, T) = 1)$*. For step 1(b), note that any hypotheses* $\mathsf{Hot}(X, 2)$ *(or more instantiated) are no longer potentially future w.r.t.* $\tau = 3$*.*

$$
\begin{aligned}
\mathcal{B}_3 = \{&\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt3}, T := 4], \{\mathsf{Hot}(\mathsf{wt3},2)\}, \{\mathsf{Hot}(\mathsf{wt3},3), \mathsf{Hot}(\mathsf{wt3},4)\}\rangle, && \textit{(1a)}\\
&\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt1}, T := 3], \{\mathsf{Hot}(\mathsf{wt1},1), \mathsf{Hot}(\mathsf{wt1},2)\}, \{\mathsf{Hot}(\mathsf{wt1},3)\}\rangle, && \textit{(1b)}\\
&\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt1}, T := 4], \{\mathsf{Hot}(\mathsf{wt1},2)\}, \{\mathsf{Hot}(\mathsf{wt1},3), \mathsf{Hot}(\mathsf{wt1},4)\}\rangle, && \textit{(1b)}\\
&\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt2}, T := 2], \{\mathsf{Hot}(\mathsf{wt2},0)\}, \{\mathsf{Hot}(\mathsf{wt2},1), \mathsf{Hot}(\mathsf{wt2},2)\}\rangle, && \textit{(1b)}\\
&\langle \mathsf{Shdn}(X,T), [X := \mathsf{wt1}, T := 2], \{\mathsf{Hot}(\mathsf{wt1},0), \mathsf{Hot}(\mathsf{wt1},1), \mathsf{Hot}(\mathsf{wt1},2)\}, \emptyset\rangle, && \textit{(1b)}\\
&\langle \mathsf{OK}(X,T), [T := 0], \{\neg\mathsf{Shdn}(X,1)\}, \emptyset\rangle, && \textit{(1b)}\\
&\langle \mathsf{OK}(X,T), [T := 1], \emptyset, \{\neg\mathsf{Shdn}(X,2)\}\rangle, && \textit{(1b)}\\
&\langle \mathsf{OK}(X,T), [T := 2], \emptyset, \{\neg\mathsf{Shdn}(X,3)\}\rangle, && \textit{(1b)}\\
&\langle \mathsf{OK}(X,T), [T := 3], \emptyset, \{\neg\mathsf{Shdn}(X,4)\}\rangle, && \textit{(2)}\\
&\langle \mathsf{Shdn}(X,T), [T := 3], \emptyset, \{\mathsf{Hot}(X,1), \mathsf{Hot}(X,2), \mathsf{Hot}(X,3)\}\rangle\} && \textit{(2)}
\end{aligned}
$$

For step 3 we consider the topological ordering $\mathsf{Shdn}_2 < \mathsf{Shdn}_3 < \mathsf{OK}_0 < \mathsf{OK}_1 < \mathsf{OK}_2 < \mathsf{OK}_3$*. When processing* Shdn_2*, there is now an answer* $[X := \mathsf{wt1}, T := 2]$*, which triggers a change in the answer for* Q_{OK}*: we need to consider all minimal substitutions that make* $\mathsf{Shdn}(X, 2)$ *ununifiable with* $\mathsf{Shdn}(\mathsf{wt1}, 2)$ *– these are* $[X := \mathsf{wt2}]$ *and* $[X := \mathsf{wt3}]$*. Therefore, in step 3(a) for* Shdn_2 *we replace the tuple*

$$\langle \mathsf{OK}(X,T), [T := 1], \emptyset, \{\neg\mathsf{Shdn}(X,2)\}\rangle$$

with the two tuples

$$
\begin{aligned}
&\langle \mathsf{OK}(X,T), [X := \mathsf{wt2}, T := 1], \emptyset, \{\neg\mathsf{Shdn}(\mathsf{wt2},2)\}\rangle\\
&\langle \mathsf{OK}(X,T), [X := \mathsf{wt3}, T := 1], \emptyset, \{\neg\mathsf{Shdn}(\mathsf{wt3},2)\}\rangle
\end{aligned}
$$

The next interesting change happens when processing OK_1*: there is a schematic hypothetical answer unifying with* $\mathsf{Shdn}(\mathsf{wt2}, 2)$*, but none unifying with* $\mathsf{Shdn}(\mathsf{wt3}, 2)$*. Therefore the latter of the previous tuples is updated in step 3(c), with this literal moving to evidence.*

At the end of this stage, we obtain:

$$\begin{aligned}\mathcal{S}_3 = \{&\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt3}, T := 4], \{\mathsf{Hot}(\mathsf{wt3},2)\}, \{\mathsf{Hot}(\mathsf{wt3},3), \mathsf{Hot}(\mathsf{wt3},4)\}\rangle,\\
&\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt1}, T := 3], \{\mathsf{Hot}(\mathsf{wt1},1), \mathsf{Hot}(\mathsf{wt1},2)\}, \{\mathsf{Hot}(\mathsf{wt1},3)\}\rangle,\\
&\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt1}, T := 4], \{\mathsf{Hot}(\mathsf{wt1},2)\}, \{\mathsf{Hot}(\mathsf{wt1},3), \mathsf{Hot}(\mathsf{wt1},4)\}\rangle,\\
&\langle \mathsf{Shdn}(X,T), [X = \mathsf{wt2}, T := 2], \{\mathsf{Hot}(\mathsf{wt2},0)\}, \{\mathsf{Hot}(\mathsf{wt2},1), \mathsf{Hot}(\mathsf{wt2},2)\}\rangle,\\
&\langle \mathsf{Shdn}(X,T), [X := \mathsf{wt1}, T := 2], \{\mathsf{Hot}(\mathsf{wt1},0), \mathsf{Hot}(\mathsf{wt1},1), \mathsf{Hot}(\mathsf{wt1},2)\}, \emptyset\rangle,\\
&\langle \mathsf{OK}(X,T), [T := 0], \{\neg\mathsf{Shdn}(X,1)\}, \emptyset\rangle,\\
&\langle \mathsf{OK}(X,T), [X := \mathsf{wt2}, T := 1], \emptyset, \{\neg\mathsf{Shdn}(\mathsf{wt2},2)\}\rangle,\\
&\langle \mathsf{OK}(X,T), [X := \mathsf{wt3}, T := 1], \{\neg\mathsf{Shdn}(\mathsf{wt3},2)\}, \emptyset\rangle,\\
&\langle \mathsf{OK}(X,T), [T := 2], \emptyset, \{\neg\mathsf{Shdn}(X,3)\}\rangle,\\
&\langle \mathsf{OK}(X,T), [T := 3], \emptyset, \{\neg\mathsf{Shdn}(X,4)\}\rangle,\\
&\langle \mathsf{Shdn}(X,T), [T := 3], \emptyset, \{\mathsf{Hot}(X,1), \mathsf{Hot}(X,2), \mathsf{Hot}(X,3)\}\rangle\}\,.\end{aligned}$$

We leave it up to the reader to verify that the remaining hypothesis in the schematic hypothetical answer $\langle \mathsf{OK}(X,T), [X := \mathsf{wt2}, T := 1], \emptyset, \{\neg\mathsf{Shdn}(\mathsf{wt2},2)\}\rangle$ *turns into evidence at time point* $\tau = 4$*, since* $D|_4 = \emptyset$ *and this is the last chance for the missing fact* $\mathsf{Hot}(\mathsf{wt2},1)$ *to arrive at the datastream.* ◁

We can also use this example to illustrate the need for step 2. Suppose that $\mathsf{Hot}(\mathsf{wt2},0)$, $\mathsf{Hot}(\mathsf{wt2},1)$ and $\mathsf{Hot}(\mathsf{wt2},2)$ were all delivered at time point 3 (i.e., they all appeared in $D|_3$). If the schematic hypothetical answer $\langle \mathsf{Shdn}(X,T), [T := 2], \emptyset, \{\mathsf{Hot}(X,0), \mathsf{Hot}(X,1), \mathsf{Hot}(X,2)\}\rangle$ had not been added to $\mathcal{B}_2$, the algorithm would incorrectly conclude $\mathsf{OK}(\mathsf{wt2},1)$ in step 3 at that point in time. The restriction for only inspecting negative hypotheses with timestamps lower than the current time may delay the production of some answers (in our example, we could have concluded $\mathsf{OK}(X,0)$ much earlier), but it is necessary to ensure that only finitely many schematic hypothetical answers are added in step 2.

Correspondence

We now discuss the relationship between the sets computed by this algorithm and the semantics defined in the previous section.

Proposition 22.

Let Π *be a positive program. For any query* $Q = \langle P, \Pi\rangle$*, substitution* θ *and sets of grounded literals* E *and* H*, if* $E \neq \emptyset$ *or the temporal argument in* $P\theta$ *is at most* τ*, then* $\langle P, \theta, E, H\rangle \in \mathcal{S}^{\downarrow}_{\tau}$ *iff* $\langle P, \theta, E, H\rangle$ *is an instance of an element of* $\mathcal{S}_\tau$.

Proof. By induction on τ. The thesis trivially holds for $\tau = -1$.

For the induction step, we note that $\mathcal{S}^{\downarrow}_{\tau+1} = \mathcal{A}_{\tau+1}$ and $\mathcal{S}_{\tau+1} = \mathcal{B}_{\tau+1}$, as there are no negated literals in Π.

Consider an arbitrary element of $\mathcal{A}_{\tau+1}$. From the definition of this set, this is a tuple of the form $\langle P, \theta, E \cup (H^+ \cap D|_{\tau+1}), H \setminus D|_{\tau+1}\rangle$ such that $\langle P, \theta, E, H\rangle \in \mathcal{S}^{\downarrow}_{\tau}$ for some $\langle P, \Pi \rangle \in \mathsf{Q}$ and $H^+ \setminus D|_{\tau+1}$ only contains future-possible atoms w.r.t. $\tau + 1$. There are three cases.

- The timestamp in $P\theta$ is at most τ. By induction hypothesis, we know that $\langle P, \theta, E, H \rangle$ is an instance of an element of $\mathcal{S}_\tau$. By definition of local mgu, $\langle P, \theta, E \cup (H^+ \cap D|_{\tau+1}), H \setminus D|_{\tau+1} \rangle$ is an instance of an element added to $\mathcal{B}_{\tau+1}$ in step 1(b).
- The timestamp in $P\theta$ is exactly $\tau + 1$ and $H^+ \cap D|_{\tau+1}$ is non-empty. Similarly to the previous case, this tuple must be an instance of an element added to $\mathcal{B}_{\tau+1}$, but now in step 1(a), by completeness of pre-processing.
- The timestamp in $P\theta$ is exactly $\tau + 1$ and $H^+ \cap D|_{\tau+1}$ is empty. Then this tuple is necessarily an instance of an element added to $\mathcal{B}_{\tau+1}$ in step 2.

For the converse, consider an element of $\mathcal{B}_{\tau+1}$. There are again three cases.

- If this element is added in step 1(a), then it is of the form $\langle P, \theta\sigma, H\sigma \cap D|_\tau, H\sigma \setminus D|_\tau \rangle$ with $\langle \theta, H \rangle \in \mathcal{P}_{\langle P, \Pi \rangle}$. By soundness of pre-processing, every ground instance $\langle P, \theta, \emptyset, H \rangle$ that only contains future-possible atoms must be in $\mathcal{S}^{\downarrow}_\tau$, so by construction of $\mathcal{A}_{\tau+1}$ any ground instance of $\langle P, \theta\sigma, H\sigma \cap D|_\tau, H\sigma \setminus D|_\tau \rangle$ must be in this set.
- If this element is added in step 1(b), then it is of the form $\langle P, \theta\sigma, E \cup E', H\sigma \setminus D|_\tau \rangle$, where $\langle P, \theta, E, H \rangle \in \mathcal{S}_\tau$ and $E' = H\sigma \cap D|_\tau$. By induction hypothesis, every ground instance of $\langle P, \theta, E, H \rangle$ is an element of $\mathcal{S}^{\downarrow}_\tau$, so every ground instance of $\langle P, \theta\sigma, E \cup E', H\sigma \setminus D|_\tau \rangle$ will also be in $\mathcal{A}_{\tau+1}$ by construction.
- If this element is added in step 2, the argument is similar to (and simpler than) the previous case. □

In the general case, the correspondence between $\mathcal{S}^{\downarrow}_\tau$ and $\mathcal{S}_\tau$ is not as precise, since evaluation of negated hypotheses is "delayed" in $\mathcal{S}_\tau$ w.r.t. $\mathcal{S}^{\downarrow}_\tau$. This means that $\mathcal{S}_\tau$ may include negated hypotheses that are already proven or contradicted in $\mathcal{S}^{\downarrow}_\tau$. Proposition 22 still holds for any queries on predicates at the lowest level of the stratification of $\Pi^{\downarrow}$ (i.e. those predicates that do not depend negatively on any other predicates). Using induction, it can be shown that an answer appears in $\mathcal{S}^{\downarrow}_\tau$ iff it is a ground instance of an answer appearing in $\mathcal{S}_{\tau'}$ for some $\tau' \geq \tau$; conversely, any ground instances of an answer in $\mathcal{S}_{\tau'}$ must appear in $\mathcal{S}^{\downarrow}_\tau$ for some $\tau \leq \tau'$. Bounds on τ' can also be determined for each concrete answer, depending on the maximum timestamp and delay for all the predicate symbols that the query depends on. Combining this observation with Corollary 6 and Proposition 7, we conclude that every answer produced by the algorithm is an answer to the original query, and every answer to the original query is eventually produced by the algorithm.

Complexity

The complexity of the algorithm for computing $\mathcal{S}_{\tau+1}$ from $\mathcal{S}_\tau$ is high, as is usual when reasoning with Datalog programs (especially in the presence of negation). We follow the standard practice of distinguishing between the complexity that arises from the structure of the actual query (including the underlying program), also called *program complexity*, and that arising from the size of the data set used for reasoning, known as *data complexity* [2, 55]. Throughout this discussion we write $|A|$ for the number of elements in a set A.

We first look into the program complexity. Step 1 of the algorithm requires computing all possible local mgus between a set H obtained from pre-processing the query and the data stream. Since every node of the corresponding SLD-tree can give rise to a local mgu (as in the proof of Lemma 21), the maximum number of local mgus grows with the number of subsets of H, in the case that all elements of H can be unified with $D|_\tau$. A more precise bound can be obtained by considering the number $|\mathsf{var}(H)|$ of distinct variables occurring in the elements of H and the number k of their possible instantiations when unifying with elements of D_τ. This yields a maximum of $(k+1)^{|\mathsf{var}(H)|}$ local mgus, which depends exponentially on $|\mathsf{var}(H)|$ – which is ultimately determined by the program's size and structure. The total number of sets H to be considered, in turn, depends linearly on the size of the different sets $\mathcal{P}_Q$ (which is related to the structure and size of the program) and $|\mathcal{S}_\tau|$.

Step 2 takes constant time on $|\mathcal{P}_Q|$, which again grows with the size of the program. Finally, Step 3 iterates over a topological sort of the predicate symbols with timestamp lower than τ, and thus requires at most $n \times \tau$ iterations, where n is the number of predicate symbols in Π. Step 3(b) iterates over all elements of the set L and $\mathcal{B}_\tau$ (both of which have size bounded by $|\mathcal{S}_\tau|$), and considers all possible substitutions with a given property. The complexity of this step is therefore bounded by $|\mathcal{S}_\tau|^2 \times |\mathsf{var}(\theta)|^c$, where θ is the substitution in the element being updated and c is the number of constants in Π. Again there is an exponential dependency on the size of Π. Step (c) includes again a quadratic loop, but with constant execution time for each iteration.

We now briefly discuss the data complexity of our algorithm. The only point in the algorithm where the data stream is used directly is Step 1, where $D|_\tau$ determines the base of the exponential factor $(k+1)^{|\mathsf{var}(H)|}$. The actual value of k is in the worst case $|(D|_\tau)| \times m$, where m is the highest arity of a predicate in Π. The size of the data stream also indirectly affects the complexity of Step 3, as it contributes to the size of $\mathcal{S}_\tau$.

This informal analysis shows that the high complexity of reasoning stems from the actual program, as is common in Datalog. This makes it more likely that our algorithm can work in practice. Indeed, the exponential factor $(k+1)^{|\mathsf{var}(H)|}$ in Step 1 is likely not too high: $|\mathsf{var}(H)|$ depends heavily on the structure of Π, and can reasonably be expected to be low (since programs are written by hand, and therefore tend not to use too many variables or make very deeply-nested arguments), while k is 0 if the elements of H are already instantiated.

The number of iterations of Step 3 can also be bounded further by observing that typical programs have a finite reasoning window (the longest possible interval between deciding the first and last literals of evidence for an answer). If this has value w, then the number of iterations will at most be $n \times w$, eliminating the dependency on τ. The bottleneck is the need to consider all possible substitutions with some property in Step 3(b); implementing this step efficiently will likely require a bit of further work into how to represent these substitutions more abstractly (for example, as sets of constraints on variables).

The original framework for hypothetical reasoning over data streams without communication delays has been implemented and applied to identifying potential trending topics in social media [21], and confirmed these expectations: the Datalog programs that were written for this specific real-world application were relatively small as expected, meaning that the exponential factors in the theoretical complexity became relatively small constants once the program was fixed. The resulting prototype was able to handle very large sets of incoming data quickly.

7 Related work and discussion

This work contributes to the field of stream reasoning, the task of conjunctively reasoning over streaming data and background knowledge [54].

Research advances on Complex Event Processors and Data Stream Management Systems [22], together with Knowledge Representation and the Semantic Web, all contributed to the several stream reasoning languages, systems and mechanisms proposed during the last decade [24].

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, 32, 33, 47], on truth maintenance systems [10], or window oriented [29], 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 [25, 34], namely those that view negation in logic programs as hypotheses and relate it to contradiction avoidance [4, 27]. 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 [46], but with a rather different approach and in the context of plan interpretation. 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 [36]. 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 [43] questioned the way those notions were defined, exploring an alternative way of looking at incomplete databases that dates back to Reiter [51], 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 [30, 39, 40, 50] 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 [41]. 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 [42].

Commercial systems [35] 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 also a concern for Walega et al. [56], who present a sound and complete stream reasoning algorithm 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 [6, 38], which was proposed for ontology-based access to temporal log data [14]. 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, Walega et al. 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.

Other authors have considered languages using negation. Our definition of stratification is similar to the concept of temporal stratification [58]. However, this notion requires the strata to be also ordered according to time; we make no such assumption in this work. A different notion of temporal stratification for stream reasoning was introducted by Beck et al. [11], but their framework also includes explicit temporal operators, making the whole formalization more complex.

8 Conclusions

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 original definitions [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 [31], or techniques to produce compact representations of search trees [48].

The algorithm we propose needs to store significant amounts of information. This was already a problem in the original framework [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 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 strictly 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.

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, Lecture Notes in Computer Science, 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. LNCS, vol. 6088, pp. 1–15. Springer (2010)

[9] Barendregt, H.P.: The lambda calculus – its syntax and semantics, Studies in

logic and the foundations of mathematics, vol. 103. North-Holland (1985)
[10] 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)
[11] Beck, H., Dao-Tran, M., Eiter, T., Fink, M.: LARS: A logic-based framework for analyzing reasoning over streams. In: Bonet and Koenig [13], pp. 1431–1438
[12] Ben-Ari, M.: Mathematical Logic for Computer Science. Springer, 3rd edn. (2012)
[13] Bonet, B., Koenig, S. (eds.): 29th AAAI Conference on Artificial Intelligence, AAAI 2015, Austin, TX, USA. AAAI Press (2015)
[14] 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)
[15] 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)
[16] Chomicki, J., Imielinski, T.: Temporal deductive databases and infinite objects. In: Edmondson-Yurkanan, C., Yannakakis, M. (eds.) Procs. PODS. pp. 61–73. ACM (1988)
[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] Cruz-Filipe, L., Gaspar, G., Nunes, I.: Can you answer while you wait? In: Varzinczak, I. (ed.) Procs. FoIKS. Lecture Notes in Computer Science, vol. 13388, pp. 111–129. Springer (2022). https://doi.org/10.1007/978-3-031-11321-5_7
[19] Cruz-Filipe, L., Gaspar, G., Nunes, I.: Reconciling communication delays and negation. In: Seidl, H., Liu, Z., Pasareanu, C.S. (eds.) Procs. ICTAC. Lecture Notes in Computer Science, vol. 13572, pp. 151–169. Springer (2022). https://doi.org/10.1007/978-3-031-17715-6_11
[20] Cruz-Filipe, L., Gaspar, G., Nunes, I.: Hypothetical answers to continuous queries over data streams. ACM Trans. Comput. Logic **25**(4), 1–40 (2024). https://doi.org/10.1145/3688845
[21] Cruz-Filipe, L., Kostopoulou, S., Montesi, F., Vistrup, J.: μxl: Explainable lead generation with microservices and hypothetical answers. In: Papadopoulos, G.A., Rademacher, F., Soldani, J. (eds.) Procs. ESOCC. Lecture Notes in Computer Science, vol. 14183, pp. 3–18. Springer (2023). https://doi.org/10.1007/978-3-031-46235-1_1
[22] 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)
[23] Dao-Tran, M., Eiter, T.: Streaming multi-context systems. In: Sierra, C. (ed.) Procs. IJCAI. pp. 1000–1007. ijcai.org (2017)
[24] 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)
[25] 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, Part I. LNCS, vol. 2407, pp. 402–436. Springer (2002)
[26] Denecker, M., Marek, V.W., Truszczynski, M.: Uniform semantic treatment of default and autoepistemic logics. Artif. Intell. **143**(1), 79–122 (2003).

https://doi.org/10.1016/S0004-3702(02)00293-X

[27] Dung, P.M.: Negations as hypotheses: An abductive foundation for logic programming. In: Furukawa, K. (ed.) Procs. ICLP. pp. 3–17. MIT Press (1991)

[28] Fitting, M.: Bilattices are nice things. In: Bolander, T., Hendricks, V., Pedersen, S.A. (eds.) Self-Reference. CSLI Publications (2006)

[29] 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)

[30] Gray, A.J., Nutt, W., Williams, M.H.: Answering queries over incomplete data stream histories. IJWIS **3**(1/2), 41–60 (2007)

[31] Guo, H., Gupta, G.: Dynamic reordering of alternatives for definite logic programs. Comput. Lang. Syst. Struct. **35**(3), 252–265 (2009)

[32] Gupta, A., Mumick, I.S., Subrahmanian, V.: Maintaining views incrementally. In: Buneman, P., Jajodia, S. (eds.) Procs. SIGMOD. pp. 157–166. ACM Press (1993)

[33] Hu, P., Motik, B., Horrocks, I.: Optimised maintenance of Datalog materialisations. In: McIlraith and Weinberger [45], pp. 1871–1879

[34] Inoue, K.: Hypothetical reasoning in logic programs. J. Log. Program. **18**(3), 191–227 (1994)

[35] 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)

[36] Jr., W.L.: On semantic issues connected with incomplete information databases. ACM Trans. Database Syst. **4**(3), 262–296 (1979)

[37] Kolaitis, P.G.: The expressive power of stratified programs. Inf. Comput. **90**(1), 50–66 (1991)

[38] Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems **2**(4), 255–299 (1990)

[39] 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)

[40] de Leng, D., Heintz, F.: Approximate stream reasoning with metric temporal logic under uncertainty. In: AAAI2019 [1], pp. 2760–2767

[41] 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)

[42] 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)

[43] 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)

[44] Lloyd, J.W.: Foundations of Logic Programming. Springer (1984)

[45] McIlraith, S.A., Weinberger, K.Q. (eds.): 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, LA, USA. AAAI Press (2018)

[46] Meadows, B.L., Langley, P., Emery, M.J.: Seeing beyond shadows: Incremental abductive reasoning for plan understanding. In: Procs. PLAN. AAAI Workshops,

vol. WS-13-13. AAAI (2013)

[47] Motik, B., Nenov, Y., Piro, R.E.F., Horrocks, I.: Incremental update of Datalog materialisation: the backward/forward algorithm. In: Bonet and Koenig [13], pp. 1560–1568

[48] Nishida, N., Vidal, G.: A framework for computing finite SLD trees. J. Log. Algebraic Methods Program. **84**(2), 197–217 (2015)

[49] 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)

[50] 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)

[51] 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)

[52] Ronca, A., Kaminski, M., Grau, B.C., Motik, B., Horrocks, I.: Stream reasoning in Temporal Datalog. In: McIlraith and Weinberger [45], pp. 1941–1948

[53] Stonebraker, M., Çetintemel, U., Zdonik, S.B.: The 8 requirements of real-time stream processing. SIGMOD Record **34**(4), 42–47 (2005)

[54] 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)

[55] Vardi, M.Y.: The complexity of relational query languages (extended abstract). In: Lewis, H.R., Simons, B.B., Burkhard, W.A., Landweber, L.H. (eds.) Procs. STOC14. pp. 137–146. ACM (1982). https://doi.org/10.1145/800070.802186

[56] Walega, P.A., Kaminski, M., Grau, B.C.: Reasoning over streaming data in Metric Temporal Datalog. In: AAAI2019 [1], pp. 3092–3099

[57] 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)

[58] Zaniolo, C.: Expressing and supporting efficiently greedy algorithms as locally stratified logic programs. In: Vos, M.D., Eiter, T., Lierler, Y., Toni, F. (eds.) Technical Communications of ICLP. CEUR Workshop Proceedings, vol. 1433. CEUR-WS.org (2015)