Reconciling communication delays and negation*

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

[1] Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark – `lcfilipe@gmail.com`

[2] LASIGE, Department of Informatics, Faculty of Sciences, University of Lisbon, Lisbon, Portugal – `{mdgaspar,minunes}@fc.ul.pt`

Abstract. Hypothetical continuous query answering over data streams was introduced as a way to anticipate answers to queries that depend on facts that may or may not happen in the future. Previous work has studied this problem for Temporal Datalog with negation and instantaneous communication, showing that hypothetical answers can be incrementally updated as new data arrives at the data stream.

In practice, individual communications take variable amounts of time, so data may arrive delayed and unordered. This motivates studying hypothetical continuous query answering in a setting with communication delays. The interaction between communication delays and negation is however problematic, and the existing approach is restricted to the positive fragment of the language. In this work we show how to remove this restriction by defining an appropriate operational semantics based on fixpoint theory, and showing that the relevant fixpoints can be computed in finite time by a carefully designed algorithm.

1 Introduction

The world of today is a constant stream of information, and the world of reasoning is no exception. Current reasoning systems are expected to receive data constantly (e.g., from sensors) and react to it in real time, continuously producing results in an online fashion. This task is known as *continuous query answering.*

One of the mainstream approaches to continuous query answering [7, 15, 34, 37, 40] considers reasoners based on logic-programming style rules depending on facts that arrive through a *data stream* (an abstract conceptualization of the inflow of information), and applies logic-based methods to compute answers to those queries.

These approaches to continuous query answering all suffer from one drawback: they only produce output once something is guaranteed to be true. In some applications – for example, if answers to queries correspond to system malfunctions – it is interesting to be able to have information about possibilities, rather than certainties, as this information can be used to foresee undesirable events.

* Partially supported by FCT through the LASIGE Research Unit, ref. UIDB/00408/2020 and ref. UIDP/00408/2020.

This observation led to the introduction of *hypothetical answers* to continuous queries [12]: answers that are consistent with (and even supported by) the information provided until now by the data stream, but that require additional future facts to be proven. Furthermore, for programs without negation, hypothetical answers can be computed by a polynomial online algorithm with an offline pre-processing step.

Most previous work on stream reasoning makes strong assumptions on the data stream – typically, that it is *ordered*: information about a given time point t is only produced after all information about previous points in time has been output. For a theoretical development, this is equivalent to assuming instantaneous communication (by disregarding the actual arrival time); in practice, such a constraint is not easy to implement, and may delay the whole system – as it requires waiting long enough to know that there can be no more information about time points previous to t lingering in the system [4, 16, 38].

Removing the assumption of an ordered data stream is tricky, since typical strategies for continuous query answering immediately break down. In the context of logical approaches to continuous query answering, the possibility of working with data that arrives out-of-order was considered in [13]. This work showed how hypothetical query answering could be addressed in a scenario with variable communication delays, but only for a language without negation – the strategy for addressing communication delays directly conflicted with the treatment of negation in [12]. Even with this restriction, the online step of the algorithm for computing hypothetical answers is no longer polynomial.

In the present work, we reconcile communication delays and negation. We show that it is possible to define an operational semantics for hypothetical continuous query answering by adapting previous definitions, accommodating for both communication delays and negation. This operational semantics is defined as the least fixpoint of a monotonic operator over a suitably defined bilattice [19], where the orders in each component reflect the two different ways in which negative information can affect hypothetical answers. The complexity stems mainly from the fact that default negation is non-monotonic, but hypothetical answers are by nature monotonic: we essentially 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). Our procedure for incrementally computing hypothetical answers in the presence of delays capitalizes on flexible strategies that deal with information as it arrives while acknowledging the possibility that older data may still arrive later on. The only requirement is that a limit is known to how delayed the information may be, which we argue is reasonable in many practical applications.

Structure. Section 2 revisits the syntax of Temporal Datalog [37] and the main ideas behind the formalism of hypothetical answers [13] in the presence of communication delays. Section 3 defines the declarative semantics of hypothetical answers for communication delays in the presence of negation. Section 4 is the bulk of our contribution, showing how an operational semantics for hypothetical answers can be defined by using a fixpoint construction. Section 5 presents the adapted online algorithm for computing this operational semantics under suitable assumptions. Section 6 discusses related work and concludes.

2 Background

This article builds heavily on previous work. In this section, we summarize the key concepts that are relevant for understanding our contribution.

2.1 Continuous queries over datastreams in Temporal Datalog

The language we work with is *Temporal Datalog* extended with negation, which is obtained from Datalog by adding the special temporal sort from [10]. Our formalism for writing continuous queries over datastreams closely follows that from [37], with only minor modifications.

Temporal Datalog extends Datalog [9] by allowing constants and variables to have two sorts: *object* or *temporal.* Sorts carry over 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. We define atoms, rules, facts and programs as usual, and assume rules to be safe: each variable in the head must occur in the body. A term, atom, rule, or program is *ground* if it contains no variables. In particular, all facts are ground. We write $\mathsf{var}(\alpha)$ for the set of variables occurring in an atom α, and extend this function homomorphically to rules and sets.

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

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

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

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

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

A substitution θ is an *answer* to query Q over Π and D if $\Pi \cup D \models Q\theta$.

We model communication delays by means of a function δ that maps each ground EDB atom in the language of Π to a natural number. The intuition is: if $\delta(P(t_1,\ldots,t_n)) = d$ and $P(t_1,\ldots,t_n) \in D|_\tau$, then $t_n \leq \tau \leq t_n + d$. We assume throughout this article that all datasets and delays satisfy this property. Function δ is extended 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 $\delta(P) = \delta(P(X_1,\ldots,X_n))$. Furthermore, we assume that $\delta(P) < \infty$ for every predicate symbol P – this is trivially the case if the delay cannot depend on the timestamp, which is a reasonable assumption in many practical scenarios.

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

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

Two high temperature readings in a row should activate the cooling system on the corresponding turbine; if the temperature remains high, there has been a malfunction, and the turbine shuts down. If the turbine does not shutdown, then we can conclude that it was working properly two time steps previously.

Suppose that there are two turbines $\mathsf{wt2}$ and $\mathsf{wt4}$, and that we know that communications from turbine $\mathsf{wt2}$ take at most 1 time units, while those from turbine $\mathsf{wt4}$ can take up to 2. Then we have e.g. $\delta(\mathsf{Temp}(\mathsf{wt2},\mathsf{high},2)) = 1$, $\delta(\mathsf{Temp}(\mathsf{wt4},X,Y)) = 2$, and $\delta(\mathsf{Temp}) = 2$. ◁

2.2 Hypothetical answers to continuous queries

Hypothetical answers to continuous queries were introduced in [12] as a means to identify potential future answers to queries – substitutions that can become answers depending on data that may yet arrive, and in particular are compatible with the available information. Furthermore, if the dataset already contains facts without which a substitution would not be an answer, then we call the corresponding hypothetical answer *supported*. We present the definitions from [13], as the original work did not consider the possibility of communication delays. These definitions only apply to the positive fragment of the language; we extend them to include negation in Section 3.

Definition 1. *A* hypothetical answer *to query Q over Π and D_τ is a pair $\langle\theta, H\rangle$, where θ is a substitution and H is a finite set of ground EDB atoms (the hypotheses) such that:*

– $\mathsf{supp}(\theta) = \mathsf{var}(Q)$*, i.e., θ only changes variables that occur in Q;*

- *H only contains* future-possible *atoms for τ, i.e., atoms for which $\tau < t_n + \delta(P(t_1, \ldots, t_n))$;*
- *$\Pi \cup D_\tau \cup H \models Q\theta$;*
- *H is minimal with respect to set inclusion.*

If the minimal subset E of D_τ such that $\Pi \cup E \cup H \models Q\theta$ is non-empty, then $\langle \theta, H, E\rangle$ is a supported answer *to Q over Π and D_τ.*

Example 2. We illustrate these concepts with Example 1. Assume that $D|_0 = \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0)\}$ and $D|_1 = \emptyset$, and let $\theta = [X := \mathsf{wt2}, T := 2]$. Then

$$\langle \theta, \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 2)\}\rangle$$

is a hypothetical answer to query $Q_E = \langle \Pi_E, \mathsf{Shdn}(X, T)\rangle$ over D_1, reflecting the intuition that $\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1)$ may still arrive in $D|_2$. This answer is supported by $\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0)$. $\triangleleft$

2.3 Operational semantics

We begin by presenting the operational semantics for hypothetical answers by means of an online algorithm with pre-processing as defined in [13] – that is, without considering the possibility of negation. We assume the reader to have some familiarity with the operational semantics of logic programming.

Pre-processing. The pre-processing step applies SLD-resolution to the program and the query Q until it reaches a goal containing only EDB atoms, and returns a set 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.

Pre-processing can be shown to terminate under some assumptions [12], which we do not discuss here. Soundness and completeness of pre-processing state that:

- if $\langle \theta, H\rangle \in P_Q$, then there exist a dataset D and substitution σ such that $Q\theta\sigma$ is ground, $H\theta \subseteq \bigcup D$ 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 $\langle \theta, H\rangle \in P_Q$ such that $\sigma = \theta\rho$ for some ρ and $H\rho \subseteq \bigcup D$.

Example 3. In the context of Example 2, pre-processing query Q_E yields the singleton set

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

It is straightforward to check that this set has the properties stated above. $\triangleleft$

Online step. The online part of the algorithm maintains a set of schematic hypothetical answers $\mathcal{S}_\tau$, where τ is the current timestamp, of the form $\langle\theta, E, H\rangle$, where E is a set of evidence.

Definition 2. *An atom $P(t_1, \ldots, t_n)$ is a* potentially future atom wrt τ *if either t_n contains a temporal variable or t_n is ground and $\tau < t_n + \delta(P(t_1, \ldots, t_n))$.*

This notion is an operational counterpart to the concept of future-possible atom, generalizing it to possibly non-ground atoms. In particular, any atom whose temporal parameter contains a variable is potentially future – intuitively, because it can be instantiated to a future timestamp.

Definition 3. *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 $\theta = \sigma\rho$.*

Local mgus were introduced to handle communication delays. Intuitively, a local mgu unifies a set of hypotheses with a subset of the datastream – leaving the possibility that some hypotheses may be instantiated at a later point in time. Local mgus can be computed by SLD-resolution [13].

Definition 4. *The set $\mathcal{S}_\tau$ of* schematic supported answers *for query Q at time τ is defined as follows.*

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

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

Example 4. We illustrate this mechanism in the setting of Example 2, where

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

We start by setting $\mathcal{S}_{-1} = \{\langle\emptyset, \emptyset, H\rangle\}$. Since $D|_0 = \{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 0)\}$, the local mgus for H and $D|_0$ are $\emptyset$ and $[X := \mathsf{wt2}, T := 0]$. Therefore,

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

Next, $D|_1 = \emptyset$, so the empty substitution is the only local mgu of H_0 and $D|_1$. Furthermore, H_0 only contains potentially future atoms wrt 1 because $\delta(\mathsf{Temp}(\mathsf{wt2}, X, Y) = 1$. The same argument applies to $D|_1$ and H, so $\mathcal{S}_1 = \mathcal{S}_0$.

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

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

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

– Finally, if $\{\mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt2}, \mathsf{high}, 2)\} \subseteq D|_2$, then $H_2 \setminus D|_2 = \emptyset$, and the system can output the answer $[X := \mathsf{wt2}, T := 0]$ to the original query. In this case, this answer (with no hypotheses) would be added to $\mathcal{S}_2$, and then trivially copied to all subsequent $\mathcal{S}_\tau$. ◁

2.4 Adding negation

The original work on hypothetical answers [12] considers a language with negation (but without communication delays). We briefly recap the concepts that we reuse from that work, and summarize the intuitions that will reappear in the current development.

Pre-processing a program whose rule bodies can contain negative literals is a generalization of the previous construction. Negative literals are not allowed to be selected during the SLD-derivation, and as a consequence they may also appear in the leaves of derivations. For each such literal $\neg \mathsf{P}(\mathsf{t}_1, \ldots, t_n)$ a fresh auxiliary query $\langle \Pi, P(X_1, \ldots, X_n)\rangle$ is generated by replacing all terms with variables.[4] 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 5. Consider again the program from Example 1, where we now consider the query $Q_{OK} = \langle \Pi_E, \mathsf{OK}(X, T)\rangle$.

Pre-processing Q_{OK} yields $P_{Q_{OK}} = \{\langle \emptyset, \neg \mathsf{Shdn}(X, T + 2)\rangle\}$, generating the auxiliary query Q_E from Example 2. Pre-processing the latter is as in Example 3; in particular it generates no fresh auxiliary queries, so we are done. ◁

Given a set of literals A, we write A^+ for the subset of positive literals in A, and A^- for the set of negative literals in A.

The online step of the algorithm for the scenario without communication delays is presented in detail in [11]. We do not discuss it here, as it has to be adapted for the current setting. The relevant aspects are explained later (Section 5). Its termination is only proved for programs that are T-stratified.

Definition 5. *Let Π be a program and let $\Pi^\downarrow$ be the program obtained from P by grounding every rule in P and replacing every atom $P(t_1, \ldots, t_{n-1}, t)$ with $P_t(t_1, \ldots, t_{n-1})$. (So every predicate symbol P in Π generates a family of predicate symbols $\{P_n \mid n \in \mathbb{N}\}$ in $\Pi^\downarrow$.)*

Program Π is T-stratified *if $\Pi^\downarrow$ is stratified (in the usual sense).*

[4] See [11] for a discussion on alternative strategies for generating the auxiliary queries. The approach used here has the advantage that there is at most one query for each predicate symbol, which simplifies the presentation.

3 Declarative semantics of negation with delays

To define a declarative semantics for hypothetical answers, we introduce the following data structure. We assume a fixed set of queries $\mathsf{Q} = \{Q_i\}_{i \in I}$, obtained by pre-processing a particular query Q_0 over a program Π, and a τ-history D_τ.

Definition 6. *A datastream D is a* possible extension of D_τ*, $D \Supset D_\tau$, if D's τ-history is exactly D_τ.*

In particular, any elements of D that do not appear in D_τ are necessarily future-possible wrt τ.

Definition 7. *A* generalized hypothetical answer *to Q over a τ-history D_τ is a family $\mathcal{S}$ of tuples $\langle Q, \theta, E, H\rangle$ where:*

- *$Q \in \mathsf{Q}$;*
- *θ is a closed substitution ranging over the variables in Q;*
- *E and H are disjoint sets of EDB atoms and negated IDB atoms such that $E^+ \subseteq D_\tau$ and all elements of H^+ are future-possible wrt τ;*
- *for every $\langle Q, \theta, E, H\rangle \in \mathcal{S}$ and $D \Supset D_\tau$, (i) $\Pi \cup D \models E^-$ and (ii) if $\Pi \cup D \models H$, then $\Pi \cup D \models Q\theta$.*

This notion differs from hypothetical answers defined in previous work in two ways: it allows negated IDB atoms in hypotheses and evidence; and it is "flattened" in the sense that it contains tuples whose first component is the query they relate to, rather than being a family indexed on queries. These options simplify our development.

4 Operational semantics of negation with delays

Our goal is to 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. In general, these sets are infinite – in the next section we discuss how to compute finite representations of them.

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

$\mathcal{S}^{\downarrow}_{\tau+1}$ is defined 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 $\langle Q, \theta, E \cup (H^+ \cap D|_{\tau+1}), H \setminus D|_{\tau+1}\rangle$ such that $\langle Q, \theta, E, H\rangle \in \mathcal{S}^{\downarrow}_\tau$ for some $Q \in \mathsf{Q}$ and $H^+ \setminus D|_{\tau+1}$ only contains future-possible atoms wrt $\tau + 1$.

The second step constructs $\mathcal{S}^{\downarrow}_{\tau+1}$ by updating the sets of negative hypotheses and evidence in $\mathcal{A}_{\tau+1}$. This is the first key contribution of this work, and the remainder of this section is dedicated to showing how it can be obtained as a fixpoint of a suitably defined operator.

4.1 The evidence lattice

As a first step, we construct a lattice over the set $\mathfrak{L}$ of all sets $\mathcal{X}$ of tuples $\langle Q, \theta, E, H\rangle$ where E and H are disjoint and such that (i) if $\langle Q, \theta, E, H\rangle \in \mathcal{X}$, then there exists $\langle Q, \theta, E', H'\rangle \in \mathcal{A}_{\tau+1}$ with $E \cup H = E' \cup H'$ and $E' \subseteq E$ and (ii) if $\langle Q, \theta, E, H\rangle$ and $\langle Q, \theta, E', H'\rangle$ are distinct elements of $\mathcal{X}$, then $E \cup H \neq E' \cup H'$. Throughout this and the next subsection we write simply $\mathcal{A}$ for $\mathcal{A}_{\tau+1}$, as this set is fixed.

Property (i) states that $\mathcal{X}$ corresponds to updating $\mathcal{A}$ with some learned evidence. Property (ii) states that this update is unique. Note that there is a one-to-one correspondence between elements of $\mathcal{A}$ and elements of $\mathcal{X}$; below, we refer to the "element of $\mathcal{A}$ generating" a tuple in $\mathcal{X}$.

Definition 8. *We define an order relation on $\mathfrak{L}$ as follows: $\mathcal{X} \sqsubseteq \mathcal{Y}$ if for every tuple $\langle Q, \theta, E, H\rangle \in \mathcal{X}$ there exists a tuple $\langle Q, \theta, E', H'\rangle \in \mathcal{Y}$ 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 1. *The relation $\sqsubseteq$ is a partial order.*

Proof. Reflexivity and transitivity are straightforward.

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

Lemma 2. *Every subset of $\mathfrak{L}$ has a least upper bound.*

Proof. Let $\mathfrak{D}$ be a subset of $\mathfrak{L}$. We claim that $\bigvee \mathfrak{D}$ is the set of all $\langle Q, \theta, E^\vee, H^\vee\rangle$ such that:

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

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

Let $\mathcal{D} \in \mathfrak{D}$ and $\langle Q, \theta, E, H\rangle \in \mathcal{D}$. Since $\langle Q, \theta, E^\vee, H^\vee\rangle \in \bigvee \mathfrak{D}$ 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 $\mathcal{Y}$ is an upper bound of $\mathfrak{D}$. We need to show that $\bigvee \mathfrak{D} \sqsubseteq \mathcal{Y}$. For this, choose $\langle Q, \theta, E^\vee, H^\vee\rangle \in \bigvee \mathfrak{D}$. For every $\langle Q, \theta, E, H\rangle \in \bigcup \mathfrak{D}$ that is generated by the same element of $\mathcal{A}$ as $\langle Q, \theta, E^\vee, H^\vee\rangle$, there must exist

$\langle Q, \theta, E', H' \rangle \in \mathcal{Y}$ 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 $\mathfrak{D}$. $\square$

Corollary 1. *$\mathfrak{L}$ is a complete lattice, which we call the* evidence lattice.

4.2 The negation update operator

The operator we consider is not defined over the evidence lattice, but over a bilattice of which $\mathfrak{L}$ is a projection.

Definition 9. *The set* $\mathbf{S}$ *contains all pairs $\langle \mathcal{X}, \mathcal{Y} \rangle$ where $\mathcal{X} \subseteq \mathcal{A}$ and $\mathcal{Y} \sqsupseteq \mathcal{A}$. We define an ordering over* $\mathbf{S}$ *by $\langle \mathcal{X}, \mathcal{Y} \rangle \preceq \langle \mathcal{X}', \mathcal{Y}' \rangle$ if $\mathcal{X} \supseteq \mathcal{X}'$ and $\mathcal{Y} \sqsubseteq \mathcal{Y}'$.*

Lemma 3. *$\langle \mathbf{S}, \preceq \rangle$ is a complete lattice.*

Proof. The result follows from the fact that both projections of $\mathbf{S}$ with the corresponding relations are complete lattices. $\square$

Definition 10. *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 $\langle Q, \theta, E, H \rangle$ for which there exists an element $\langle Q', \theta', E', \emptyset \rangle \in \mathcal{Y}$ such that $\neg Q'\theta \in H$.*
- *$R_2(\mathcal{X}, \mathcal{Y})$ is obtained from $\mathcal{Y}$ by replacing every tuple $\langle Q, \theta, E, H \rangle$ with the tuple $\langle Q, \theta, E \cup P, H \setminus P \rangle$ where P is the set of all $\neg\alpha$ such that there exists no tuple $\langle Q', \theta', E', H' \rangle \in \mathcal{X}$ with $\alpha = Q'\theta'$.*

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 is essential to proving that we can reach a fixpoint.

Lemma 4. *R is monotonic.*

Proof. Assume 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 $R_2(\mathcal{X}, \mathcal{Y}) \sqsubseteq R_2(\mathcal{X}', \mathcal{Y}')$.

For the first, observe that the result of R_1 is computed by removing some tuples from its first argument. Since $\mathcal{X} \supseteq \mathcal{X}'$, it suffices to show that any tuples removed from $\mathcal{X}$ are also removed from $\mathcal{X}'$. A tuple $\langle Q, \theta, E, H \rangle$ is removed from $\mathcal{X}$ if there exists $\langle Q', \theta', E', \emptyset \rangle \in \mathcal{Y}$ such that $\neg Q'\theta \in H$. Since $\mathcal{Y} \sqsubseteq \mathcal{Y}'$, there must be a tuple $\langle Q', \theta', E'', H'' \rangle \in \mathcal{Y}'$ such that $E'' \cup H'' = E' \cup \emptyset = E'$ and $H'' \subseteq \emptyset$. These conditions imply that $H'' = \emptyset$ and $E'' = E'$, i.e. $\langle Q', \theta', E', \emptyset \rangle \in \mathcal{Y}'$, and therefore $\langle Q, \theta, E, H \rangle$ is also removed from $\mathcal{X}'$ when computing $R_1(\mathcal{X}', \mathcal{Y}')$.

For the second, we note that the result of R_2 is computed by moving some literals from H to E in some tuples $\langle Q, \theta, E, H\rangle$ in its second argument. Since $\mathcal{Y} \sqsubseteq \mathcal{Y}'$, for each such tuple in $\mathcal{Y}$ there must be $\langle Q, \theta, E', H'\rangle$ such that $E \cup H = E' \cup H'$ and $E \subseteq E'$. The thesis can then be established by showing that any literals moved from H to E that are not already in E' will also be moved from H' to E'. Now, these literals are of the form $\neg\alpha$ and such that there exists no tuple $\langle Q^*, \theta^*, E^*, H^*\rangle \in \mathcal{X}$ with $\alpha = Q^*\theta^*$. Since $\mathcal{X} \supseteq \mathcal{X}'$, there can also be no such tuple in $\mathcal{X}'$, and therefore the same literal must also be moved from H' to E' when computing $R_2(\mathcal{X}', \mathcal{Y}')$, unless it already is in E' to start with.

Therefore R is a monotonic operator. $\square$

Corollary 2. *R has a least fixpoint.*

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

Let $\langle \mathcal{X}_0, \mathcal{Y}_0\rangle$ be the least fixpoint of R, and define $\mathcal{S}^{\downarrow}_{\tau+1}$ as the set of tuples $\langle Q, \theta, E, H\rangle \in \mathcal{Y}_0$ for which there exists $\langle Q, \theta, E', H'\rangle \in \mathcal{X}_0$ with $E \cup H = E' \cup H'$. For convenience, we write simply $\mathcal{S}^{\downarrow}$ for $\mathcal{S}^{\downarrow}_{\tau+1}$ in the remainder of this section. Intuitively, $\mathcal{S}^{\downarrow}$ contains the tuples from $\mathcal{A}$ that remain in $\mathcal{X}_0$, with their sets of evidence updated as in $\mathcal{Y}_0$. 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 5. *$\langle \mathcal{S}^{\downarrow}, \mathcal{S}^{\downarrow}\rangle$ 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 $i = 1, 2$.

$R_1(\mathcal{S}^{\downarrow}, \mathcal{S}^{\downarrow})$ is obtained by removing from $\mathcal{S}^{\downarrow}$ the tuples $\langle Q, \theta, E, H\rangle$ for which there exists $\langle Q', \theta', E', \emptyset\rangle \in \mathcal{S}^{\downarrow}$ such that $\neg Q'\theta \in H$. But such a tuple $\langle Q', \theta', E', \emptyset\rangle$ 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 $\langle Q, \theta, E, H\rangle \in \mathcal{S}^{\downarrow}$ by moving literals of the form $\neg\alpha \in H$ to E if there exists no tuple $\langle Q', \theta', E', H'\rangle \in \mathcal{S}^{\downarrow}$ with $\alpha = Q'\theta'$. 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. $\square$

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

4.3 Soundness and completeness

We begin this section with a simple lemma.

Lemma 6. *If $\langle Q, \theta, E, H \rangle \in \mathcal{S}^{\downarrow}_{\tau}$ or $\langle Q, \theta, E, H \rangle \in \mathcal{A}_{\tau}$, then $E^+ \subseteq D_\tau$ and every element of H^+ is future-possible wrt τ.*

Proof. Straightforward by induction on τ. □

We now proceed to show that each $\mathcal{S}^{\downarrow}_{\tau}$ indeed is a generalized hypothetical answer to Q over D_τ. This is done by proving the following property by induction on τ:

> Assume that $D \Supset D_\tau$. Then $\langle Q, \theta, E, H \rangle \in \mathcal{S}^{\downarrow}_{\tau}$ 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 $\Pi \cup D \models Q\theta$ and (ii) $\mathcal{D}$ is an SLD$^\neg$-derivation proving that $\Pi \cup E \cup H \models Q\theta$ that uses all elements of $E \cup H$. $(*)$

By SLD$^\neg$-derivation we simply mean a (normal) SLD-derivation where negated atoms are treated by checking whether they appear as facts (in our case, in 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 a straightforward consequence of how pre-processing is defined; the interested reader can find a proof in [11].

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

Proof. Suppose that $D \Supset D_{\tau+1}$. Then also $D \Supset D_\tau$.

For the direct implication, assume that $\langle Q, \theta, E, H \rangle \in \mathcal{A}_{\tau+1}$ is such that $H^+ \subseteq D \setminus D_{\tau+1}$. By construction of $\mathcal{A}_{\tau+1}$, there exists $\langle Q, \theta, E', H' \rangle \in \mathcal{S}^{\downarrow}_{\tau}$ 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 $\Pi \cup D \models Q\theta$ and $\mathcal{D}$ is an SLD$^\neg$-derivation proving that $\Pi \cup E' \cup H' \models Q\theta$ 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 $\Pi \cup E \cup H \models Q\theta$ that uses all elements of $E \cup H$.

For the converse implication, let $\mathcal{D}$ be an SLDNF-derivation proving that $\Pi \cup D \models Q\theta$, 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 $\Pi \cup F \models Q\theta$. By hypothesis on $\mathcal{S}^{\downarrow}_{\tau}$, there exists a tuple $\langle Q, \theta, E, H \rangle \in \mathcal{S}^{\downarrow}_{\tau}$ such that $H^+ \subseteq D \setminus D_\tau$ and $E \cup H = F$.

If $\langle Q, \theta, E, H \rangle \notin \mathcal{A}_{\tau+1}$, then $H^+ \setminus D_{\tau+1}$ contains some elements that are not future-possible wrt $\tau+1$; 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 $\mathcal{A}_{\tau+1}$ contains an element $\langle Q, \theta, E', H' \rangle$ 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 $\mathcal{A}_{\tau+1}$. We start with an auxiliary definition.

Definition 11. *For $\langle \mathcal{X}, \mathcal{Y} \rangle \in \mathbf{S}$, define $\mathcal{X} \sqcap \mathcal{Y}$ to be the set of tuples $\langle Q, \theta, E, H \rangle \in \mathcal{Y}$ for which there exists $\langle Q, \theta, E', H' \rangle \in \mathcal{X}$ with $E \cup H = E' \cup H'$, $E^+ = (E')^+$ and $H^+ = (H')^+$.*

In particular, $\mathcal{S}^{\downarrow}_{\tau+1} = \mathcal{X}_0 \sqcap \mathcal{Y}_0$, where $\langle \mathcal{X}_0, \mathcal{Y}_0 \rangle$ is the least fixpoint of R.

The proof uses transfinite induction. For simplicity, we again split it in several lemmas. Note that 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 8. *Let $\langle \mathcal{X}, \mathcal{Y} \rangle \in \mathbf{S}$ be such that: (i) if $\langle Q, \theta, E, H \rangle \in \mathcal{X} \sqcap \mathcal{Y}$, 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. Assume that $D \Supset D_{\tau+1}$.

For the direct implication, choose $\langle Q, \theta, E, H \rangle \in R_1(\mathcal{X}, \mathcal{Y}) \sqcap R_2(\mathcal{X}, \mathcal{Y})$. In particular $\langle Q, \theta, E, H \rangle \in R_2(\mathcal{X}, \mathcal{Y})$, so there exists a tuple $\langle Q, \theta, E', H' \rangle \in \mathcal{X} \sqcap \mathcal{Y}$ 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 $\Pi \cup D \models Q\theta$ and define F as before 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 $\langle Q, \theta, E, H \rangle \in \mathcal{X} \sqcap \mathcal{Y}$ such that $H^+ \subseteq (D \setminus D_{\tau+1})$ and $E \cup H = F$. By definition of $\sqcap$, there exists a tuple $\langle Q, \theta, E_X, H_X \rangle \in \mathcal{X}$ such that $E_X \cup H_X = E \cup H = F$, and $\langle Q, \theta, E, H \rangle \in \mathcal{Y}$. By definition of R_2, there is also a tuple $\langle Q, \theta, E_Y, H_Y \rangle \in R_2(\mathcal{X}, \mathcal{Y})$ such that $E_Y \cup H_Y = E \cup H = F$ and $H_Y^+ = H^+$. To establish the thesis, we only need to show that $\langle Q, \theta, E_X, H_X \rangle \in R_1(\mathcal{X}, \mathcal{Y})$.

Assume that this is not the case. Then there exists an element $\langle Q', \theta', E', \emptyset \rangle \in \mathcal{Y}$ with $\neg Q'\theta' \in H_X$. 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 $\Pi \cup D \models Q\theta'$, which contradicts the fact that $\mathcal{D}$ at some point must process the fact $\neg Q'\theta'$ by showing that no such derivation exists. □

Lemma 9. *Let $\{\langle \mathcal{X}_i, \mathcal{Y}_i \rangle \mid i \in I\}$ be a directed subset of $\mathbf{S}$ such that $\mathcal{X}_i \sqcap \mathcal{Y}_i$ satisfies $(*)$ for all $i \in I$. Let $\langle \mathcal{X}, \mathcal{Y} \rangle = \bigvee \{\langle \mathcal{X}_i, \mathcal{Y}_i \rangle \mid i \in I\}$. Then $\mathcal{X} \sqcap \mathcal{Y}$ satisfies $(*)$.*

Proof. Assume that $D \Supset D_{\tau+1}$.

For the direct implication, pick $\langle Q, \theta, E, H \rangle \in \mathcal{X} \sqcap \mathcal{Y}$ with $H^+ \subseteq D \setminus D_{\tau+1}$. Then $\langle Q, \theta, E, H \rangle \in \mathcal{Y}_i$ for all i, and every $\mathcal{X}_i$ contains an element $\langle Q, \theta, E_i, H_i \rangle$ 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 $\Pi \cup D \models Q\theta$ and define F again as above. By hypothesis, for each i there must exist $\langle Q, \theta, E_i, H_i \rangle \in \mathcal{X}_i \sqcap \mathcal{Y}_i$ with $E_i \cup H_i = F$ and such that $H_i^+ \subseteq D \setminus D_{\tau+1}$. This means that

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

Corollary 3. *If $\mathcal{A}_{\tau+1}$ satisfies $(*)$, then $\mathcal{S}^{\downarrow}_{\tau+1}$ also satisfies $(*)$.*

Theorem 1. *If $\langle Q, \theta, E, H \rangle \in \mathcal{S}^{\downarrow}_{\tau}$, then $\langle Q, \theta, E, H \rangle$ is a generalized hypothetical answer to Q over D_τ.*

Proof. Let $\langle Q, \theta, E, H \rangle \in \mathcal{S}^{\downarrow}_{\tau}$. By construction, $Q \in \mathsf{Q}$ and θ is a closed substitution ranging over the variables in Q: these properties were guaranteed in the construction of $\mathcal{S}^{\downarrow}_{-1}$, and Q and θ are never changed afterwards. Furthermore, due to the way E and H are constructed and updated, they are necessarily disjoint and contain only EDB atoms and negated IDB atoms: this is true for $\mathcal{S}^{\downarrow}_{-1}$ by construction (where $E = \emptyset$), and all later updates are of the form "move an element from H to E", which preserves these properties.

E^+ and H^+ are only changed in the construction of $\mathcal{A}_\tau$, where elements are added to E^+ if they appear in $D|_\tau$ (so induction guarantees $E^+ \subseteq D_\tau$), and the tuple is only kept if all elements remaining in H^+ are future-possible wrt τ.

For the last point, we recall that $\mathcal{S}^{\downarrow}_{\tau}$ satisfies $(*)$. Assume that $D \Supset D_\tau$.

To show that $\Pi \cup D \models E^-$, assume towards a contradiction that this is not the case, and choose an element $\neg\alpha \in E^-$ such that $\Pi \cup D \models \alpha$. Due to how pre-processing works, there exist a query Q' and a substitution θ' such that $\alpha = Q'\theta'$; furthermore, θ' is closed, and it can be assumed to range exactly over the variables in Q'. By completeness of SLDNF-resolution, there is an SLDNF-derivation $\mathcal{D}$ showing that $\Pi \cup D \models Q'\theta'$. Take F to be the elements of D that are used in $\mathcal{D}$ and all negative literals that appear in $\mathcal{D}$. Then $\mathcal{D}$ is again an SLD$^\neg$-derivation proving that $\Pi \cup F \models Q'\theta'$, whence $\mathcal{S}^{\downarrow}_{\tau}$ must contain a tuple $\langle Q', \theta', E', H' \rangle$ with $E' \cup H' = F$. But this contradicts the hypothesis that $\neg\alpha \in E^-$, since the only way to add elements to E^- is by application of R (specifically, R_2) when no such tuple exists, and the construction of $\mathcal{S}^{\downarrow}_{\tau}$ ensures that all tuples it contains ultimately originate from $\mathcal{S}^{\downarrow}_{-1}$.

Lastly, assume that $\Pi \cup D \models H$. Then $H^+ \subseteq (D \setminus D_\tau)$, from which property $(*)$ ensures existence of an SLDNF-derivation $\mathcal{D}$ proving that $\Pi \cup D \models Q\theta$. $\square$

5 Computing the operational semantics

The operational semantics in the previous section is based on infinite sets, and can therefore not be directly implemented. In this section, we show that we can represent generalized hypothetical answers finitely and update them one timestamp at a time. We argue informally that our construction is correct.

We extend the algorithm from [13], which deals with communication delays in the positive fragment of our language (see Section 2.3), with the ideas from [11] for dealing with negation. 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 schematic hypothetical answers for every query that may need to be examined when updating negations are introduced, 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. Furthermore, we assume that our program is T-stratified.

The algorithm is as follows. We initialize $\mathcal{S}_{-1}(Q) = \emptyset$ for every query Q.

1. *Define a set $\mathcal{B}_\tau$ updating $\mathcal{S}_{\tau-1}$ with information from the datastream.*
   (a) If $\langle \theta, H \rangle \in \mathcal{P}(Q)$ and σ is a non-empty local mgu for H and $D|_\tau$ such that $H\sigma \setminus D|_\tau$ only contains potentially future atoms wrt τ, then $\langle \theta\sigma, E \cup E', H\sigma \setminus D|_\tau \rangle \in \mathcal{B}_\tau(Q)$, where $E' = H\sigma \cap D|_\tau$.
   (b) If $\langle \theta, E, H \rangle \in \mathcal{S}_{\tau-1}(Q)$ 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 wrt τ, then $\langle \theta\sigma, E \cup E', H\sigma \setminus D|_\tau \rangle \in \mathcal{B}_\tau(Q)$, where $E' = H\sigma \cap D|_\tau$.
2. *Add answers to queries that might be examined when updating negations.*
   For each query $Q \in \mathsf{Q}$, let $\sigma = [T := \tau]$ with T the temporal variable in Q. If $\langle \theta, H \rangle \in \mathcal{P}_Q$ and every element of $H\sigma$ is either potentially future wrt τ or negated, then add $\langle \theta\sigma, \emptyset, H\sigma \rangle$ to $\mathcal{B}_\tau(Q)$.
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 Q_P has at least one schematic answer with $T\theta = t$ (where Q_P is the query on P and T is the temporal parameter in Q_P). For each of these P_t in order:
   (a) set $\ell = P(t_1, \ldots, t_n)$ and let $S(P_t)$ be the set of elements $\langle \theta, E, H \rangle$ in $\mathcal{S}_\tau(Q_P)$ such that $T\theta = t$;
   (b) if $S(P_t)$ contains a tuple $\langle \theta, E', \emptyset \rangle$, then: in each $\mathcal{B}_\tau(Q)$, replace every $\langle \sigma, E, H \rangle$ such that $\ell\sigma$ is unifiable with an element $h \in H^-$ with all possible $\langle \sigma\theta', E\theta', H\theta' \rangle$ such that θ' is a minimal substitution with the property that $\ell\theta$ is not unifiable with $h\theta'$;
   (c) for each $\langle \theta, E, H \rangle \in \mathcal{B}_\tau(Q)$ for some Q, if H^- contains an element h with predicate symbol P and timestamp $t \leq \tau$ and there is no tuple $\langle \theta', E', H' \rangle \in S(P_t)$ such that h and $\ell\theta'$ are unifiable, then remove $\neg h$ from H and add it to E.

Step (1) is essentially the update step from [13]. The use of local mgus ensures that no answers are lost even in presence of communication delays. Step (2), adapted from [11], 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. The sets $\mathcal{B}_\tau(Q)$ obtained at the end of this step correspond (modulo instantiation) to the subset of elements $\langle Q, \theta, E, H \rangle \in \mathcal{A}_\tau$ where either $E \neq \emptyset$ or $T\theta \leq \tau$. Step (3), also adapted from [11], updates negated hypotheses. The final result of this step again corresponds, modulo instantiation, to the subset of elements of $\mathcal{S}_\tau$ with the same property as above.

6 Related work and discussion

This work contributes to the field of stream reasoning, the task of conjunctively reasoning over streaming data and background knowledge [39].

Research advances on Complex Event Processors and Data Stream Management Systems [14], together with Knowledge Representation and the Semantic Web, all contributed to the several stream reasoning languages, systems and mechanisms proposed during the last decade [16].

Computing answers to a query over a data source that is continuously producing information requires techniques with some kind of *incremental evaluation*, in order to avoid reevaluating the query from scratch each time new information arrives. Efforts in this direction have capitalized on incremental algorithms based on seminaive evaluation [1, 5, 22, 23, 33], based on truth maintenance systems [6], or window oriented [20], among others. Being an incremental variant of SLD-resolution, our framework [12, 13] fits naturally in the first class.

Hypothetical query answering over streams is broadly related to abduction in logic programming [17, 24], namely to approaches that view negated atoms as hypotheses and relate them to contradiction avoidance [2, 18]. In this sense, we apply an incremental form of data-driven abductive inference similar to [32], but with a different approach and in a different context. To our knowledge, hypothetical or abductive reasoning has not been previously applied to continuous query answering, although it has been applied to annotation of stream data [3].

Also incomplete databases include notions of possible and certain answers [25]. Here, possible answers are answers to a complete database D' that the incomplete database D can represent, while certain answers belong to all complete databases that D can represent. Libkin [30] explored an alternative way of looking at incomplete databases that dates back to Reiter [36], viewing a database as a logical theory. He explored the semantics of incompleteness independently of a particular data model, appealing to orderings to describe the degree of incompleteness of a database. Other authors [21, 26, 27, 35] 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 from the area of databases and event processing have developed techniques to deal with out-of-order data. An example is inserting special marks in the input stream (punctuations) to guide window processing [28], which assert a timestamp that is a lower bound for all future incoming values of an 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 [29].

Other authors have considered languages using negation. Our definition of stratification is similar to the concept of temporal stratification from [41]. 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 is given in [7], but their framework also includes explicit temporal operators, making the whole formalization more complex.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Alferes, J.J., Pereira, L.M.: Reasoning with Logic Programming, Lecture Notes in Computer Science, vol. 1111. Springer (1996)
3. Alirezaie, M., Loutfi, A.: Automated reasoning using abduction for interpretation of medical signals. J. Biomed. Semant. **5**, 35 (2014)
4. 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)
5. 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)
6. 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)
7. Beck, H., Dao-Tran, M., Eiter, T., Fink, M.: LARS: A logic-based framework for analyzing reasoning over streams. In: Bonet and Koenig [8], pp. 1431–1438
8. Bonet, B., Koenig, S. (eds.): 29th AAAI Conference on Artificial Intelligence, AAAI 2015, Austin, TX, USA. AAAI Press (2015)
9. 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)
10. Chomicki, J., Imielinski, T.: Temporal deductive databases and infinite objects. In: Edmondson-Yurkanan, C., Yannakakis, M. (eds.) Procs. PODS. pp. 61–73. ACM (1988)
11. Cruz-Filipe, L., Gaspar, G., Nunes, I.: Hypothetical answers to continuous queries over data streams. CoRR **abs/1905.09610** (2019), submitted for publication.
12. Cruz-Filipe, L., Gaspar, G., Nunes, I.: Hypothetical answers to continuous queries over data streams. In: Procs. AAAI. pp. 2798–2805. AAAI Press (2020)
13. Cruz-Filipe, L., Gaspar, G., Nunes, I.: Can you answer while you wait? In: Procs. FoIKS. LNCS, Springer (2022), to appear.
14. 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)
15. Dao-Tran, M., Eiter, T.: Streaming multi-context systems. In: Sierra, C. (ed.) Procs. IJCAI. pp. 1000–1007. ijcai.org (2017)
16. 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)
17. 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)
18. Dung, P.M.: Negations as hypotheses: An abductive foundation for logic programming. In: Furukawa, K. (ed.) Procs. ICLP. pp. 3–17. MIT Press (1991)
19. Fitting, M.: Bilattices are nice things. In: Bolander, T., Hendricks, V., Pedersen, S.A. (eds.) Self-Reference. CSLI Publications (2006)
20. 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)

21. Gray, A.J., Nutt, W., Williams, M.H.: Answering queries over incomplete data stream histories. IJWIS **3**(1/2), 41–60 (2007)
22. Gupta, A., Mumick, I.S., Subrahmanian, V.: Maintaining views incrementally. In: Buneman, P., Jajodia, S. (eds.) Procs. SIGMOD. pp. 157–166. ACM Press (1993)
23. Hu, P., Motik, B., Horrocks, I.: Optimised maintenance of datalog materialisations. In: McIlraith and Weinberger [31], pp. 1871–1879
24. Inoue, K.: Hypothetical reasoning in logic programs. J. Log. Program. **18**(3), 191–227 (1994)
25. Jr., W.L.: On semantic issues connected with incomplete information databases. ACM Trans. Database Syst. **4**(3), 262–296 (1979)
26. 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)
27. de Leng, D., Heintz, F.: Approximate stream reasoning with metric temporal logic under uncertainty. In: Procs. AAAI. pp. 2760–2767. AAAI Press (2019)
28. 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)
29. 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)
30. 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)
31. McIlraith, S.A., Weinberger, K.Q. (eds.): 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, LA, USA. AAAI Press (2018)
32. 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)
33. Motik, B., Nenov, Y., Piro, R.E.F., Horrocks, I.: Incremental update of datalog materialisation: the backward/forward algorithm. In: Bonet and Koenig [8], pp. 1560–1568
34. 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)
35. 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)
36. 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)
37. Ronca, A., Kaminski, M., Grau, B.C., Motik, B., Horrocks, I.: Stream reasoning in temporal datalog. In: McIlraith and Weinberger [31], pp. 1941–1948
38. Stonebraker, M., Çetintemel, U., Zdonik, S.B.: The 8 requirements of real-time stream processing. SIGMOD Record **34**(4), 42–47 (2005)
39. 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)
40. 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)

41. 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)