# Hypothetical answers to continuous queries over data streams

LUÍS CRUZ-FILIPE, University of Southern Denmark, Denmark
GRAÇA GASPAR, Faculdade de Ciências, Universidade de Lisboa, Portugal
ISABEL NUNES, Faculdade de Ciências, Universidade de Lisboa, Portugal

Answers to continuous queries over data streams are often delayed until some relevant input arrives through the data stream. These delays may turn answers, when they arrive, obsolete to users who sometimes have to make decisions with no help whatsoever. Therefore, it can be useful to provide hypothetical answers – "given the current information, it is possible that $X$ will become true at time $t$" – instead of no information at all.

In this work we present a semantics for queries and corresponding answers that covers such hypothetical answers, together with an incremental online algorithm for updating the set of facts that are consistent with the currently available information. Our framework also works in a language supporting negation.

CCS Concepts: • **Theory of computation → Constraint and logic programming**; **Automated reasoning**; Online algorithms; • **Computing methodologies → Logic programming and answer set programming**; *Temporal reasoning*.

**ACM Reference Format:**
Luís Cruz-Filipe, Graça Gaspar, and Isabel Nunes. 2024. Hypothetical answers to continuous queries over data streams. *ACM Trans. Comput. Logic* 1, 1 (August 2024), 40 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Modern-day reasoning systems often have to react to real-time information about the real world provided by e.g. sensors. This information is typically conceptualized as a data stream, which is accessed by the reasoning system. The reasoning tasks associated to data streams – usually called *continuous queries* – are expected to run continuously and produce results through another data stream in an online fashion, as new elements arrive.

A data stream is a potentially unbounded sequence of data items generated by an active, uncontrolled data source. Elements arrive continuously at the system, potentially unordered, and at unpredictable rates. Thus, reasoning over data streams requires dealing with incomplete or missing data, potentially storing large amounts of data (in case it might be needed to answer future queries), and providing answers in timely fashion – among other problems, see e.g. [Babcock et al. 2002; Dell'Aglio et al. 2017; Stonebraker et al. 2005].

The output stream is normally ordered by time, which implies that the system may have to delay appending some answer because of uncertainty in possible answers relating to earlier time points. The length of this delay may be unpredictable (*unbound wait*) or infinite, for example if the query uses operators that range over the whole input data stream (*blocking operations*). In these

Authors' addresses: Luís Cruz-Filipe, Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, 5230, Odense, Denmark, lcfilipe@gmail.com; Graça Gaspar, LASIGE, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, Campo Grande, 1749-016, Lisbon, Portugal, mdgaspar@fc.ul.pt; Isabel Nunes, LASIGE, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, Campo Grande, 1749-016, Lisbon, Portugal, minunes@fc.ul.pt.


1529-3785/2024/8-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

cases, answers that have been computed may never be output. An approach to avoid this problem is to restrict the language by forbidding blocking operations [Ronca et al. 2018a; Zaniolo 2012]. Another approach uses the concept of reasoning window [Beck et al. 2015b; Özçep et al. 2014], which bounds the size of the input that can be used for computing each output (either in time units or in number of events). In our work, we do not impose such restrictions, and we show that we are able to provide useful information even in some instances of blocking queries.

In several applications, it is useful to know that some answers are likely to be produced in the future, since there is already some information that might lead to their generation. This is the case namely in prognosis systems (e.g., medical diagnosis, stock market prediction), where one can prepare for the possibility of something happening. To this goal, we propose *hypothetical answers*: answers that are supported by information provided by the input stream, but that still depend on other facts being true in the future. Knowledge about both the facts that support the answer and possible future facts that may make it true gives users the possibility to make timely, informed decisions in contexts where preemptive measures may have to be taken.

Moreover, by giving such hypothetical answers to the user we cope with unbound wait in a constructive way, since the system is no longer "mute" while waiting for an answer to become definitive.

Many existing approaches to reasoning with data streams adapt and extend models, languages and techniques used for querying databases and the semantic web [Arasu et al. 2006; Barbieri et al. 2009]. We develop our theory in line with the works of [Beck et al. 2015b; Dao-Tran and Eiter 2017; Özçep et al. 2014; Ronca et al. 2022; Zaniolo 2012], where continuous queries are treated as rules of a logic program consisting of both rules and facts arriving through a data stream. Our techniques bear some similarites to previous work on abduction (see the discussion on related work in Section 7), but to the best of our knowledge this is the first time that hypothetical reasoning has been considered in the context of continuous query answering.

*Contribution.* We present a declarative semantics for queries in Temporal Datalog, where we define the notions of hypothetical and supported answers. We define an operational semantics based on SLD-resolution, where we compute answers with premises to queries. We then show that there is a natural connection between the answers computed by the operational semantics and the hypothetical and supported answers defined declaratively. By refining SLD-resolution, we obtain an incremental online algorithm for maintaining and updating the set of answers that are consistent with the currently available information. Finally, we define Temporal Datalog with negation and show that our results can be extended to this language.

*Publication history.* A preliminary version of this work, not including proofs, was published in conference proceedings [Cruz-Filipe et al. 2020]. The current article extends that work with proofs of all results and further examples. Section 6 (negation), which was only a sketch, was completely restructured, and now includes a precise definition of the declarative and operational semantics, as well as full proofs of all results, and several examples. Several results in this section were not stated previously, including soundness and completeness.

*Structure.* Section 2 revisits some fundamental background notions, namely the formalism from [Ronca et al. 2018b], which we extend in this article. Section 3 introduces our declarative semantics for continuous queries, defining hypothetical and supported answers, and relates these concepts with the standard definitions of answers. Section 4 presents our operational semantics for continuous queries and relates it to the declarative semantics. Section 5 details our online algorithm to compute supported answers incrementally, as input facts arrive through the data stream, and

proves it sound and complete. Section 6 extends our framework to negation by failure. Section 7 compares our proposal to similar ones in the literature, discussing its advantages.

## 2 BACKGROUND

### 2.1 Continuous queries in Temporal Datalog

We use the framework from [Ronca et al. 2018b] to write continuous queries over datastreams, slightly adapting some definitions. We work in *Temporal Datalog*, the fragment of negation-free Datalog extended with the special temporal sort from [Chomicki and Imielinski 1988], which is isomorphic to the set of natural numbers equipped with addition with arbitrary constants. In Section 6 we extend this language with negation.

*Syntax of Temporal Datalog.* A *vocabulary* consists of constants (numbers or identifiers in lowercase), variables (single uppercase letters) and predicate symbols (identifiers beginning with an uppercase letter). All these may be indexed if necessary; occurrences of predicates and variables are distinguished by context. In examples, we use words in sans serif for concrete constants and predicates.

Constants and variables have one of two sorts: *object* or *temporal*. An *object term* is either an object (constant) or an object variable. A *time term* is either a natural number (called a *time point* or *temporal constant*), a time variable, or an expression of the form $T + \mathsf{k}$ where $T$ is a time variable and $\mathsf{k}$ is an integer.

Predicates can take at most one temporal parameter, which we assume to be the last one (if present). A predicate with no temporal parameter is called *rigid*, otherwise it is called *temporal*. An atom is an expression $P(t_1, \ldots, t_n)$ where $P$ is a predicate and each $t_i$ is a term of the expected sort. Without loss of generality, we assume that all predicates are temporal. Rigid predicates can be made temporal by adding a dummy temporal parameter that does not affect their semantics. In other words, we replace every instance of $P(t_1, \ldots, t_n)$ by $P'(t_1, \ldots, t_n, 0)$. Thus, $P'(t_1, \ldots, t_n, 0)$ holds iff $P(t_1, \ldots, t_n)$ holds – and the values of $P'$ at time points other than 0 will have no impact on the semantics.

A rule has the form $\wedge_i \alpha_i \rightarrow \alpha$, where $\alpha$ and each $\alpha_i$ are temporal atoms. Atom $\alpha$ is called the *head* of the rule, and $\wedge_i \alpha_i$ the *body*. Rules are assumed to be *safe*: each variable in the head must occur in the body. A *program* is a set of rules.

If a predicate symbol occurs in an atom in the head of a rule with non-empty body, we call that predicate *intensional*, or an IDB predicate. A predicate that is defined only through rules with empty body is called *extensional*, or an EDB predicate. An atom is extensional/intensional (or EDB/IDB atom) according to whether its predicate symbol is extensional or intensional.

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

Rules are instantiated by means of *substitutions*, which are functions mapping variables to terms of the expected sort. The *support* of a substitution $\theta$ is the set $\mathrm{supp}(\theta) = \{X \mid \theta(X) \neq X\}$. We consider only substitutions with finite support, and write $\theta = [X_1 := t_1, \ldots, X_n := t_n]$ for the substitution mapping each variable $X_i$ to the term $t_i$, and leaving all remaining variables unchanged. We call each expression $X_i := t_i$ a *binding*. A substitution is *ground* if every variable in its support is mapped to a constant. An *instance* $r' = r\theta$ of a rule $r$ is obtained by simultaneously replacing every variable $X$ in $r$ by $\theta(X)$ and computing any additions of temporal constants.

A *temporal query* is a pair $Q = \langle P, \Pi \rangle$ where $\Pi$ is a program and $P$ is an IDB atom in the language underlying $\Pi$. We do not require $P$ to be ground, and typically the temporal parameter is uninstantiated.[1]

A *dataset* is an indexed family of sets of EDB facts (*input facts*), each intuitively corresponding to the facts delivered by a data stream to a reasoning system at each time point.

Definition 1. *A* dataset *is a family $D = \{D|_\tau \mid \tau \in \mathbb{N}\}$, where $D|_\tau$ is a set of EDB facts with timestamp $\tau$.*

We call $D|_\tau$ the *$\tau$-slice* of $D$. For each $\tau$, we also consider $D$'s $\tau$-history $D_\tau = \bigcup\{D|_{\tau'} \mid \tau' \leq \tau\}$.

From these definitions it follows that $D|_\tau = D_\tau \setminus D_{\tau-1}$ for every $\tau$, and that $D_\tau$ contains only facts whose temporal argument is at most $\tau$. By convention, $D_{-1} = \emptyset$.

*Semantics.* The semantics of Temporal Datalog is a variant of the standard semantics based on Herbrand models. A Herbrand interpretation $I$ for Temporal Datalog is a set of facts. If $\alpha$ is an atom with no variables, then we define $\bar{\alpha}$ as the fact obtained from $\alpha$ by evaluating its temporal term. We say that $I$ satisfies $\alpha$, $I \models \alpha$, if $\bar{\alpha} \in I$. The extension of the notion of satisfaction to the whole language follows the standard construction, and the definition of entailment is the standard one.

An *answer* to a query $Q = \langle P, \Pi \rangle$ over a dataset $D$ is a ground substitution $\theta$ whose domain is the set of variables in $P$, satisfying $\Pi \cup D \models P\theta$. In the context of continuous query answering, we are interested in the case where $D$ is a $\tau$-history of some data stream, which changes with time. We denote the set of all answers to $Q$ over $D_\tau$ as $\mathcal{A}(Q, D, \tau)$.

We illustrate the extension we propose with a Temporal Datalog program, which is a small variant of Example 1 in [Ronca et al. 2018b]. We will return to this example throughout our presentation.

Example 1. *A set of wind turbines are scattered throughout the North Sea. Each turbine is equipped with a sensor that continuously sends temperature readings* $\mathsf{Temp}(\mathit{Device}, \mathit{Level}, \mathit{Time})$ *to a data centre. The data centre tracks activation of cooling measures in each turbine, recording malfunctions and shutdowns by means of the following program $\Pi_E$.*

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

*Consider the query $Q_E = \langle \mathsf{Malf}(X, T), \Pi_E \rangle$. If $D_0 = \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0)\}$, then at time point 0 we can infer* $\mathsf{Flag}(\mathsf{wt25}, 0)$*, but there is no output for $Q_E$. If* $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1)$ *arrives to $D$ at time point 1, then $D_1 = D_0 \cup \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1)\}$, and we can infer* $\mathsf{Flag}(\mathsf{wt25}, 1)$ *and* $\mathsf{Cool}(\mathsf{wt25}, 1)$*, but there still is no answer to $Q_E$. Finally, the arrival of* $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)$ *to $D$ at time point 2 yields $D_2 = D_1 \cup \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}$, allowing us to infer* $\mathsf{Flag}(\mathsf{wt25}, 2)$*,* $\mathsf{Cool}(\mathsf{wt25}, 2)$*,* $\mathsf{Shdn}(\mathsf{wt25}, 2)$ *and* $\mathsf{Malf}(\mathsf{wt25}, 0)$*. Then $\{X := \mathsf{wt25}, T := 0\} \in \mathcal{A}(Q_E, D, 2)$.* ◁

## 2.2 SLD-resolution

We now revisit some concepts from SLD-resolution, adapted from [Ben-Ari 2012; Lloyd 1984].

A *literal* is an atom or its negation. Atoms are also called *positive* literals, and a negated atom is a *negative* literal. A *definite clause* is a disjunction of literals containing at most one positive literal. In the case where all literals are negative, the clause is a *goal*. We use the standard rule notation for writing definite clauses.

[1] The most common exception is if $P$ represents a property that does not depend on time, which normally would be written using a rigid predicate. In our framework, this would mean that $P$'s temporal parameter would be instantiated to 0.

Definition 2. *Given two substitutions* $\theta = [X_1 := t_1, \ldots, X_m := t_m]$ *and* $\sigma = [Y_1 := u_1, \ldots, Y_n := u_n]$, *their* composition *is the substitution* $\theta\sigma$ *obtained from*

$$[X_1 := t_1\sigma, \ldots, X_m := t_m\sigma, Y_1 := u_1, \ldots, Y_n := u_n]$$

*by (i) deleting any binding where* $t_i\sigma = X_i$ *and (ii) deleting any binding* $Y_j := u_j$ *where* $Y_j \in \{X_1, \ldots, X_m\}$.

Condition (i) removes redundant identity bindings. Condition (ii) ensures that, for every atom $\alpha$, $\alpha(\theta\sigma) = (\alpha\theta)\sigma$.

Definition 3. *Two atomic formulas* $P(\vec{X})$ *and* $P(\vec{Y})$ *are* unifiable *if there exists a substitution* $\theta$, *also called a* unifier, *such that* $P(\vec{X})\theta = P(\vec{Y})\theta$.

*A unifier* $\theta$ *of* $P(\vec{X})$ *and* $P(\vec{Y})$ *is called a* most general unifier (mgu) *if for each unifier* $\sigma$ *of* $P(\vec{X})$ *and* $P(\vec{Y})$ *there exists a substitution* $\gamma$ *such that* $\sigma = \theta\gamma$.

It is well known that there always exist several mgus of any two unifiable atoms, and that they are unique up to renaming of variables.

Recall that a *goal* is a clause of the form $\neg \wedge_j \beta_j$. If $C$ is a rule $\wedge_i \alpha_i \rightarrow \alpha$, $G$ is a goal $\neg \wedge_j \beta_j$ with $\mathrm{var}(G) \cap \mathrm{var}(C) = \emptyset$, and $\theta$ is an mgu of $\alpha$ and $\beta_k$, then the *resolvent* of $G$ and $C$ using $\theta$ is the goal $\neg \left(\bigwedge_{j<k} \beta_j \wedge \bigwedge_i \alpha_i \wedge \bigwedge_{j>k} \beta_j\right)\theta$.

If $\Pi$ is a program and $G$ is a goal, an *SLD-derivation* of $\Pi \cup \{G\}$ is a (finite or infinite) sequence $G_0, G_1, \ldots$ of goals with $G = G_0$, a sequence $C_1, C_2, \ldots$ of $\alpha$-renamings[2] of program clauses of $\Pi$ and a sequence $\theta_1, \theta_2, \ldots$ of substitutions such that $G_{i+1}$ is the resolvent of $G_i$ and $C_{i+1}$ using $\theta_{i+1}$. A finite SLD-derivation of $\Pi \cup \{G\}$ where the last goal $G_n$ is a contradiction ($\square$) is called an *SLD-refutation* of $\Pi \cup \{G\}$ of length $n$, and the substitution obtained by restricting the composition of $\theta_1, \ldots, \theta_n$ to the variables occurring in $G$ is called a *computed answer* of $\Pi \cup \{G\}$.

Proposition 1 (Soundness). *Let* $\Pi$ *be a program,* $G$ *be a goal, and suppose that there is an SLD-refutation of* $\Pi \cup \{G\}$ *with computed answer* $\theta$. *Then* $\Pi \models \forall(\neg G\theta)$.

Proposition 2 (Completeness). *Let* $\Pi$ *be a program,* $G$ *be a goal, and suppose that* $\Pi \models \forall(\neg G\theta)$. *Then there is an SLD-refutation of* $\Pi \cup \{G\}$ *with computed answer* $\theta$.

The notation $\forall(\neg G\theta)$ stands for the formula $\forall x_1 \ldots x_n.\neg G\theta$, where $\mathrm{var}(G\theta) = \{x_1, \ldots, x_n\}$. Since $G$ is of the form $\neg \wedge_j \beta_j$, this formula can be written equivalently as $\forall x_1 \ldots x_n.\ \wedge_j (\beta_j\theta)$ – i.e., $\Pi \models \forall(\neg G\theta)$ in the previous two theorems states that $\Pi$ entails any instance of any atom in $G$.

There are two degrees of freedom when trying to build SLD-derivations: which literal to choose in the goal at any given point, and which clause to unify it with (in case there are several possibilities). The choice of the goal is dictated by a *computation rule*; for a given computation rule $R$, the *SLD-tree* for $\Pi$ and $G$ is the tree containing $G$, and where each node contains one descendent for each possible resolvent of the literal chosen by $R$ in that node and the head of a rule in $\Pi$.

Proposition 3 (Independence of the computation rule). *Let* $\Pi$ *be a program and* $G$ *be a goal. Then every SLD-tree for* $\Pi$ *and* $G$ *has the same (finite or infinite) number of branches ending with the empty clause.*

This result states that the computation rule does not affect whether one can find an SLD-refutation for $\Pi$ and $G$.

[2]I.e. program clauses from $\Pi$ whose variables have been renamed.

## 3 HYPOTHETICAL ANSWERS

In our running example, $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0)$ being delivered at time instant 0 yields some evidence that $\mathsf{Malf}(\mathsf{wt25}, 0)$ may turn out to be true. Later, we may receive further evidence as in the example (the arrival of $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1)$), or we might find out that this fact will not be true (if $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1)$ does not arrive).

We propose a theory that also allows for such *hypothetical answers* to a continuous query: if some substitution can become an answer as long as some facts in the future are true, then we output this information. In this way we can lessen the negative effects of unbound wait. Hypothetical answers can also refer to future time points: in our example, $[X := \mathsf{wt25}, T := 2]$ would also be output at time point 0 as a substitution that may prove to be an actual answer to the query $\langle \mathsf{Shdn}(X, T), \Pi_E \rangle$ when further information arrives.

Our formalism uses ideas from multi-valued logic, where some substitutions correspond to answers (true), others are known not to be answers (false), and others are consistent with the available data, but can not yet be shown to be true or false. In our example, the fact $\mathsf{Malf}(\mathsf{wt25}, 0)$ is consistent with the data at time point 0, and thus "possible"; it is also consistent with the data at time point 1, and thus "more possible"; and it finally becomes (known to be) true at time point 2.

As already motivated, we want answers to give us not only the substitutions that make the query goal true, but also ones that make the query goal possible in the following sense: they depend both on facts that have been delivered by the data stream and on facts that still may be delivered by the data stream (i.e., those whose timestamp is greater than $\tau$).

Definition 4. *A ground atom $P(t_1, \ldots, t_n)$ is* future-possible for $\tau$ *if $\tau < t_n$.*

Definition 5. *A* hypothetical answer *to query $Q = \langle P, \Pi \rangle$ over $D_\tau$ is a pair $\langle \theta, H \rangle$, where $\theta$ is a substitution and $H$ is a finite set of ground EDB atoms (the hypotheses) such that:*

- $\mathsf{supp}(\theta) = \mathsf{var}(P)$*;*
- *$H$ only contains atoms future-possible for $\tau$;*
- *$\Pi \cup D_\tau \cup H \models P\theta$;*
- *$H$ is minimal with respect to set inclusion.*

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

Intuitively, a hypothetical answer $\langle \theta, H \rangle$ states that $P\theta$ holds if all facts in $H$ are ever delivered by the data stream. Thus, $P\theta$ is currently backed up by the information available. In particular, if $H = \emptyset$ then $P\theta$ is an answer in the standard sense (it is a known fact).

Proposition 4. *Let $Q = \langle P, \Pi \rangle$ be a query, $D$ be a data stream and $\tau$ be a time instant. If $\langle \theta, \emptyset \rangle \in \mathcal{H}(Q, D, \tau)$, then $\theta \in \mathcal{A}(Q, D, \tau)$.*

Proof. If $\langle \theta, H \rangle \in \mathcal{H}(Q, D, \tau)$, then $\Pi \cup D_\tau \cup H \models P\theta$. When $H = \emptyset$, this reduces to $\Pi \cup D_\tau \models P\theta$, which coincides with the definition of answer. □

We can generalize this proposition, formalizing the intuition we gave for the definition of hypothetical answer.

Proposition 5. *Let $Q = \langle P, \Pi \rangle$ be a query, $D$ be a data stream and $\tau$ be a time instant. If $\langle \theta, H \rangle \in \mathcal{H}(Q, D, \tau)$, then there exist a time point $\tau' \geq \tau$ and a data stream $D'$ such that $D_\tau = D'_\tau$ and $\theta \in \mathcal{A}(Q, D', \tau')$.*

Proof. Let $D'$ be the data stream such that:

- $D'|_t = D|_t$ for $t \leq \tau$;
- $D'|_t$ contains the elements of $H$ with timestamp $t$ for $t > \tau$.

Let $\tau'$ be the highest timestamp in $H$. It is straightforward to verify that $D'$ satisfies the thesis. □

Example 2. *We illustrate these concepts in the context of Example 1.*
*Consider* $\theta = [X := \mathsf{wt25}, T := 0]$. *Since* $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0) \in D|_0$, *Then*

$$\langle \theta, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}\rangle \in \mathcal{H}(Q_E, D, 0)\,.$$

*Since* $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1) \in D|_1$, *we conclude that* $\langle \theta, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}\rangle \in \mathcal{H}(Q_E, D, 1)$. *Finally,* $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2) \in D|_2$, *so* $\langle \theta, \emptyset\rangle \in \mathcal{H}(Q_E, D, 2)$. *This answer has no hypotheses, and indeed* $\theta \in \mathcal{A}(Q_E, D, 2)$.
*Now consider* $\theta' = [X := \mathsf{wt42}, T := 1]$ *for another constant* wt42. *Then* $\mathcal{H}(Q_E, D, 0)$ *also includes e.g.* $\langle \theta', \{\mathsf{Temp}(\mathsf{wt42}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt42}, \mathsf{high}, 2), \mathsf{Temp}(\mathsf{wt42}, \mathsf{high}, 3)\}\rangle$. *However, since* $D|_1$ *does not contain* $\mathsf{Temp}(\mathsf{wt42}, \mathsf{high}, 1)$, *there is no element* $\langle \theta', H'\rangle \in \mathcal{H}(Q_E, D, \tau)$ *for* $\tau \geq 1$. ◁

Hypothetical answers $\langle \theta, H\rangle \in \mathcal{H}(Q, D, \tau)$ where $H \neq \emptyset$ can be further split into two kinds: those that are supported by some fact(s) delivered by the datastream, and those for which there is no evidence whatsover – they only depend on facts whose truth value is unknown. For the former, $\Pi \cup H \not\models P\theta$: they rely on some fact from $D_\tau$. This is the class of answers that interests us, as there is non-trivial information in saying that they may become true.

Definition 6. *Let* $Q = \langle P, \Pi\rangle$ *be a query,* $D$ *be a data stream and* $\tau$ *be a time instant. A non-empty set of ground EDB atoms* $E \subseteq D_\tau$ *is* evidence *supporting* $\langle \theta, H\rangle \in \mathcal{H}(Q, D, \tau)$ *if* $E$ *is a minimal set satisfying* $\Pi \cup E \cup H \models P\theta$. *A* supported answer *to* $Q$ *over* $D_\tau$ *is a triple* $\langle \theta, H, E\rangle$ *such that* $\Pi \cup H \not\models P\theta$ *where* $E$ *is evidence supporting* $\langle \theta, H\rangle$. *We denote the set of supported answers to* $Q$ *over* $D_\tau$ *by* $\mathcal{E}(Q, D, \tau)$.

Since we are working with finite sets, if $\langle \theta, H\rangle \in \mathcal{H}(Q, D, \tau)$ and $\Pi \cup E \cup H \models P\theta$, then there exists a set $E'$ such that $\langle \theta, H, E'\rangle$ is a supported answer to $Q$ over $D_\tau$. However, in general, several such sets $E'$ may exist. As a consequence, Propositions 4 and 5 generalize to supported answers in the obvious way. The condition $\Pi \cup H \not\models P\theta$ ensures that the evidence must be used to infer the answer.

Example 3. *In Example 2, the hypothetical answer* $\langle \theta, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}\rangle$ *is supported by the evidence* $\{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0)\}$, *while* $\langle \theta, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}\rangle$ *is supported by* $\{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1)\}$.
*Since there is no evidence for* $\langle \theta', \{\mathsf{Temp}(\mathsf{wt42}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt42}, \mathsf{high}, 2), \mathsf{Temp}(\mathsf{wt42}, \mathsf{high}, 3)\}\rangle$, *this answer is not supported.* ◁

This example illustrates that unsupported hypothetical answers are not very informative: it is the existence of supporting evidence that distinguishes interesting hypothetical answers from any arbitrary future fact.

However, it is useful to consider even unsupported hypothetical answers in order to develop incremental algorithms to compute supported answers: the sequence of sets $\Theta^E_\tau = \{\theta \mid \langle \theta, H, E\rangle \in \mathcal{E}(Q, D, \tau)$ for some $H, E\}$ is not monotonic, as at every time point new unsupported hypothetical answers may get evidence and supported hypothetical answers may get rejected. The sequence $\Theta^H_\tau = \{\theta \mid \langle \theta, H\rangle \in \mathcal{H}(Q, D, \tau)$ for some $H\}$, on the other hand, is anti-monotonic, as the following results state.

Proposition 6. *Let* $Q = \langle P, \Pi\rangle$ *be a query,* $D$ *be a data stream and* $\tau$ *be a time instant. If* $\langle \theta, H\rangle \in \mathcal{H}(Q, D, \tau)$, *then there exists* $H^0$ *such that* $\langle \theta, H^0\rangle \in \mathcal{H}(Q, D, -1)$ *and* $H = H^0 \setminus D_\tau$. *Furthermore, if* $H \neq H^0$, *then* $\langle \theta, H, H^0 \setminus H\rangle \in \mathcal{E}(Q, D, \tau)$.

Proof. Recall that $D_{-1} = \emptyset$ by convention. If $\langle \theta, H \rangle \in \mathcal{H}(Q, D, \tau)$, then $\Pi \cup D_\tau \cup H \models P\theta$. Since $D_\tau \cup H$ is finite, there is a minimal subset $H^0$ of $D_\tau \cup H$ with the property that $H \subseteq H^0$ and $\Pi \cup H^0 \models P\theta$. Clearly $H = H^0 \setminus D_\tau$.

Assume that $H^- \subseteq H^0$ is also such that $\Pi \cup H^- \models P\theta$. Then $\Pi \cup D_\tau \cup (H^- \setminus D_\tau) \models P\theta$; but $\langle \theta, H \rangle \in \mathcal{H}(Q, D, \tau)$, so $H \subseteq H^- \setminus D_\tau$ and therefore also $H \subseteq H^-$. By definition of $H^0$, this implies that $H^0 \subseteq H^-$, hence $\langle \theta, H^0 \rangle \in \mathcal{H}(Q, D, -1)$.

Finally, if $E \subseteq H^0 \setminus H$ is evidence supporting $\langle \theta, H \rangle$, then $\Pi \cup E \cup H \models P\theta$, hence $H^0 \subseteq E \cup H$, since $\langle \theta, E \cup H \rangle \in \mathcal{H}(Q, D, -1)$. □

Proposition 7. *Let $Q = \langle P, \Pi \rangle$ be a query, $D$ be a data stream and $\tau$ be a time instant. If $\langle \theta, H \rangle \in \mathcal{H}(Q, D, \tau)$ and $\tau' < \tau$, then there exists $\langle \theta, H' \rangle \in \mathcal{H}(Q, D, \tau')$ such that $H = H' \setminus (D_\tau \setminus D_{\tau'})$.*

Proof. Just as the proof of the previous proposition, but dividing $\Pi \cup D_\tau$ into $\Pi \cup D_{\tau'}$ and $D_\tau \setminus D_{\tau'}$ instead of into $\Pi$ and $D_\tau$. □

Examples 2 and 3 also illustrate this property, with hypotheses turning into evidence as time progresses. Since $D_{-1} = \emptyset$, Proposition 6 is a particular case of Proposition 7.

In the next sections we show how to compute hypothetical answers and the corresponding sets of evidence for a given continuous query.

## 4 OPERATIONAL SEMANTICS VIA SLD-RESOLUTION

The definitions of hypothetical and supported answers are declarative. We now show how SLD-resolution can be adapted to algorithms that compute these answers. We use standard results about SLD-resolution, see for example [Lloyd 1984].

We begin with a simple observation: since the only function symbol in our language is addition of temporal parameters (which is invertible), we can always choose mgus that do not replace variables in the goal with new ones.

Lemma 8. *Let $\neg \wedge_i \alpha_i$ be a goal and $\wedge_j \beta_j \rightarrow \beta$ be a rule such that $\beta$ is unifiable with $\alpha_k$ for some $k$. Then there is an mgu $\theta = [X_1 := t_1, \ldots, X_n := t_n]$ of $\alpha_k$ and $\beta$ such that all variables occurring in $t_1, \ldots, t_n$ also occur in $\alpha_k$.*

Proof. Let $\rho = [X_1 := t_1, \ldots, X_n := t_n]$ be an mgu of $\alpha_k$ and $\beta$. For each $i$, $t_i$ can either be a variable $Y_i$ or a time expression $T_i + k_i$. First, iteratively build a substitution $\sigma$ as follows: for $i \in [1, \ldots, n]$, if $X_i$ occurs in $\alpha_k$ but $t_i$ does not and $\sigma$ does not yet include a replacement for the variable in $t_i$, extend $\sigma$ with $Y_i := X_i$, if $t_i$ is $Y_i$, or $T_i := X_i - k_i$, if $t_i$ is $T_i + k_i$.

We now show that $\theta = \rho\sigma$ is an mgu of $\alpha_k$ and $\beta$ with the desired property. If $X := t \in \theta$, then either (i) $X$ is $X_i$ and $t$ is $t_i\sigma$ for some $i$ or (ii) $X := t \in \sigma$. In case (i), by construction of $\sigma$ if $X_i$ occurs in $\alpha_k$ but $t_i$ includes a variable not in $\alpha_k$, then $\sigma$ replaces that variable with a term using only variables in $\alpha_k$. In case (ii), by construction $X$ does not occur in $\alpha_k$.

To show that $\theta$ is an mgu of $\alpha_k$ and $\beta$ it suffices to observe that $\sigma$ is invertible, with $\sigma^{-1} = [X := Y \mid Y := X \in \sigma]\, [X := T - k \mid T := X + k \in \sigma]$. □

Without loss of generality, we assume all mgus in SLD-derivations to have the property in Lemma 8.

Our intuition is as follows: classical SLD-resolution computes substitutions that make a conjunction of atoms a logical consequence of a program by constructing SLD-derivations that end in the empty clause. We relax this by allowing derivations to end with a goal if: this goal only refers to EDB predicates and all the temporal terms in it refer to future instants (possibly after further instantiation). This makes the notion of derivation also dependent on a time parameter.

Definition 7. *An atom $P(t_1, \ldots, t_n)$ is a* potentially future atom wrt $\tau$ *if it is future-possible for $\tau$ or either $t_n$ contains a temporal variable.*

This concept generalizes the notion of future-possible atom for $\tau$ to possibly non-ground atoms. In particular, any atom whose time parameter contains a variable is a potentially future atom, since it may be instantiated to a time point in the future.

Definition 8. *An* SLD-refutation with future premises *of $Q = \langle P, \Pi \rangle$ over $D_\tau$ is a finite SLD-derivation of $\Pi \cup D_\tau \cup \{\neg P\}$ whose last goal only contains potentially future EDB atoms wrt $\tau$.*

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

Example 4. *Consider query $Q_E$ from Example 1 and $\tau = 1$. There is an SLD-derivation of $\Pi_E \cup D_1 \cup \{\neg \mathsf{Malf}(X, T)\}$ ending with $\leftarrow \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)$, which contains a single potentially future EDB atom with respect to 1. Thus, $\langle Q_E, D_1 \rangle \vdash_{\mathsf{SLD}} \langle \theta, \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2) \rangle$ with $\theta = [X := \mathsf{wt25}, T := 0]$.* ◁

As this example illustrates, computed answers with premises are the operational counterpart to hypothetical answers, with two caveats. First, a computed answer with premises need not be ground: there may be some universally quantified variables in the last goal. Second, $\wedge_i \alpha_i$ may contain redundant conjuncts, in the sense that they might not be needed to establish the goal. We briefly illustrate these two features.

Example 5. *In our running example, there is also an SLD-derivation of $\Pi_E \cup D_1 \cup \{\neg \mathsf{Malf}(X, T)\}$ ending with the goal $\neg \bigwedge_{i=0}^{2} \mathsf{Temp}(X, \mathsf{high}, T + i)$, which only contains potentially future EDB atoms with respect to 1. Thus also $\langle Q_E, D_1 \rangle \vdash_{\mathsf{SLD}} \langle \emptyset, \bigwedge_{i=0}^{2} \mathsf{Temp}(X, \mathsf{high}, T + i) \rangle$.* ◁

Example 6. *Consider the following program $\Pi'$:*

$$\mathsf{P}(\mathsf{a}, T) \rightarrow \mathsf{R}(\mathsf{a}, T) \qquad\qquad \mathsf{P}(\mathsf{a}, T) \wedge \mathsf{Q}(\mathsf{a}, T) \rightarrow \mathsf{R}(\mathsf{a}, T)$$

*and the query $Q' = \langle \mathsf{R}(X, T), \Pi' \rangle$.*

*Let $D'|_0 = \emptyset$. Then there is an SLD-derivation of $\Pi' \cup D'_0 \cup \{\neg \mathsf{R}(X, T)\}$ that ends with the goal $\neg (\mathsf{P}(\mathsf{a}, T) \wedge \mathsf{Q}(\mathsf{a}, T))$, which only contains potentially future EDB atoms wrt $\tau$. Thus $\langle Q', D'_0 \rangle \vdash_{\mathsf{SLD}} \langle [X := \mathsf{a}], \mathsf{P}(\mathsf{a}, T) \wedge \mathsf{Q}(\mathsf{a}, T) \rangle$. However, atom $\mathsf{Q}(\mathsf{a}, T)$ is redundant, since $\mathsf{P}(\mathsf{a}, T)$ suffices to make $[X := \mathsf{a}]$ an answer to $Q$ for any $T$.*

*(Observe that also $\langle Q', D'_0 \rangle \vdash_{\mathsf{SLD}} \langle [X := \mathsf{a}], \mathsf{P}(\mathsf{a}, T) \rangle$, but produced by a different SLD-derivation.)* ◁

We now look at the relationship between the operational definition of computed answer with premises and the notion of hypothetical answer. The examples above show that these notions do not precisely correspond. However, we can show that computed answers with premises approximate hypothetical answers and, conversely, every hypothetical answer is a grounded instance of a computed answer with premises.

Theorem 1 (Soundness). *Let $Q = \langle P, \Pi \rangle$ be a query, $D$ be a data stream and $\tau$ be a time instant. Assume that $\langle Q, D_\tau \rangle \vdash_{\mathsf{SLD}} \langle \theta, \wedge_i \alpha_i \rangle$. Let $\sigma$ be a ground substitution such that: (i) $\mathsf{supp}(\sigma) = \mathsf{var}(\wedge_i \alpha_i) \cup (\mathsf{var}(P) \setminus \mathsf{supp}(\theta))$ and (ii) if $\alpha_i$ is $P(t_1, \ldots, t_n)$, then $t_n \sigma > \tau$. Then there is a set $H \subseteq \{\alpha_i \sigma\}_i$ such that $\langle (\theta\sigma)|_{\mathsf{var}(P)}, H \rangle \in \mathcal{H}(Q, D, \tau)$.*

Proof. Assume that there is some SLD-refutation with future premises of $Q$ over $D_\tau$. Then this is an SLD-derivation whose last goal $G = \neg \wedge_i \alpha_i$ only contains potentially future EDB atoms with respect to $\tau$. Let $\sigma$ be any substitution in the conditions of the hypothesis. Taking $H' = \{\alpha_i \sigma\}_i$, we can extend this SLD-derivation to a (standard) SLD-refutation for $\Pi \cup D_\tau \cup H' \cup \{\neg P\}$, by resolving $G$ with each of the $\alpha_i$ in turn. The computed answer is then the restriction of $\theta\sigma$ to $\mathsf{var}(P)$. By

soundness of SLD-resolution, $\Pi \cup D_\tau \cup H' \models P(\theta\sigma)|_{\mathrm{var}(P)}$. Since $H'$ is finite, we can find a minimal set $H \subseteq H'$ with the latter property. □

Theorem 2 (Completeness). *Let $Q = \langle P, \Pi\rangle$ be a query, $D$ be a data stream and $\tau$ be a time instant. If $\langle\theta, H\rangle \in \mathcal{H}(Q, D, \tau)$, then there exist substitutions $\rho$ and $\sigma$ and a finite set of atoms $\{\alpha_i\}_i$ such that $\theta = \rho\sigma$, $H = \{\alpha_i\sigma\}_i$ and $\langle Q, D_\tau\rangle \vdash_{\mathsf{SLD}} \langle\rho, \wedge_i\alpha_i\rangle$.*

Proof. Suppose $\langle\theta, H\rangle \in \mathcal{H}(Q, D, \tau)$. Then $\Pi \cup D_\tau \cup H \models P\theta$. By completeness of SLD-resolution, there exist substitutions $\gamma$ and $\delta$ and an SLD-derivation for $\Pi \cup D_\tau \cup H \cup \{\neg P\}$ with computed answer $\gamma$ such that $\theta = \gamma\delta$.

By minimality of $H$, for each $\alpha \in H$ there must exist a step in this SLD-derivation where the current goal is resolved with $\alpha$. By independence of the computation rule, we can assume that these are the last steps in the derivation. Let $\mathcal{D}'$ be the derivation consisting only of these steps, and $\mathcal{D}$ be the original derivation without $\mathcal{D}'$. Let $\rho$ be the answer computed by $\mathcal{D}$ and $G = \neg \wedge_i \alpha_i$ be its last goal, let $\rho'$ be the answer computed by $\mathcal{D}'$, and define $\sigma = \rho'\delta$. Then:

- Let $X \in \mathrm{supp}(\theta)$; then $X$ occurs in $P$. If $\rho(X)$ occurs in $G$, then by construction $\gamma(X) = (\rho\rho')(X)$. If $\rho(X)$ is a ground term or $\rho(X)$ does not occur in $G$, then trivially $\gamma(X) = \rho(X) = (\rho\rho')(X)$ since $\rho'$ does not change $\rho(X)$. In either case, $\theta = \gamma\delta = \rho\rho'\delta = \rho\sigma$.
- $H = \{\alpha_i\sigma\}_i$: by construction of $\mathcal{D}'$, $H = \{\alpha_i\rho'\}_i$, and since $\alpha_i\rho'$ is ground for each $i$, it is also equal to $\alpha_i\rho'\delta = \alpha_i\sigma$.

The derivation $\mathcal{D}$ shows that $\langle Q, D_\tau\rangle \vdash_{\mathsf{SLD}} \langle\rho, \wedge_i\alpha_i\rangle$. □

All notions introduced in this section depend on the time parameter $\tau$, and in particular on the history dataset $D_\tau$. In the next section, we explore the idea of "organizing" the SLD-derivation adequately to pre-process $\Pi$ independently of $D_\tau$, so that the computation of (hypothetical) answers can be split into an offline part and a less expensive online part.

## 5 INCREMENTAL COMPUTATION OF HYPOTHETICAL ANSWERS

Proposition 7 states that the set of hypothetical answers evolves as time passes, with hypothetical answers either gaining evidence and becoming query answers or being put aside due to their dependence on facts that turn out not to be true.

In this section, we show how we can use this temporal evolution to compute supported answers incrementally. We start by showing a simple result on SLD-derivations that allows us to reorganize the resolution steps based on the timestamp (§ 5.1). Using this result, we propose a two-stage process for computing and updating supported answers to continuous queries over data streams: a first (offline) stage pre-processes the program as much as possible (§ 5.2), and a second (online) stage updates a set of supported answers based on incoming information (§ 5.3). Correctness of the online stage, due to its complexity, is delegated to a separate section (§ 5.4).

### 5.1 A two-stage process

We start by revisiting SLD-derivations and showing how they can reflect the structure of the datastream.

Proposition 9. *Let $Q = \langle P, \Pi\rangle$ be a query and $D$ be a data stream. For any time constant $\tau$, if $\langle Q, D_\tau\rangle \vdash_{\mathsf{SLD}} \langle\theta, \wedge_i\alpha_i\rangle$, then there exist an SLD-refutation with future premises of $Q$ over $D_\tau$ computing $\langle\theta, \wedge_i\alpha_i\rangle$ and a sequence $k_{-1} \leq k_0 \leq \ldots \leq k_\tau$ such that:*

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

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

An SLD-refutation with future premises with the property guaranteed by Proposition 9 is called a *stratified* SLD-refutation with future premises. Since data stream $D$ only contains EDB atoms, it also follows that in a stratified SLD-refutation all goals after $G_{k_{-1}}$ are always resolved with EDB atoms. Furthermore, each goal $G_{k_i}$ contains only potentially future EDB atoms with respect to $i$. Let $\theta_i$ be the restriction of the composition of all substitutions in the SLD-derivation up to step $k_i$ to $\mathrm{var}(P)$. Then $G_{k_i} = \neg \wedge_j \alpha_j$ represents all hypothetical answers to $Q$ over $D_i$ of the form $\langle (\theta_i\sigma)|_{\mathrm{var}(P)}, \wedge_j \alpha_j \rangle$ for some ground substitution $\sigma$ (cf. Theorem 1).

This yields an online procedure to compute supported answers to continuous queries over data streams. In a pre-processing step, we calculate all computed answers with premises to $Q$ over $D_{-1}$, and keep the ones with minimal set of formulas. (Note that Theorem 2 guarantees that all minimal sets are generated by this procedure, although some non-minimal sets may also appear as in Example 5.) The online part of the procedure then resolves each of these sets with the facts delivered by the data stream, adding the resulting resolvents to a set of schemata of supported answers (i.e. where variables may still occur). By Proposition 9, if there is at least one resolution step at this stage, then the hypothetical answers represented by these schemata all have evidence, so they are indeed supported.

### 5.2 Pre-processing step

We now look at the pre-processing step of our procedure in more detail. In general, this step may not terminate, as the following example illustrates.

Example 7. *Consider the following program* $\Pi''$*, where* $\mathsf{R}$ *is an extensional predicate and* $\mathsf{S}$ *is an intensional predicate.*

$$\mathsf{S}(X,T) \rightarrow \mathsf{S}(X,T+1) \qquad\qquad \mathsf{R}(X,T) \rightarrow \mathsf{S}(X,T)$$

*If* $\mathsf{R}(\mathsf{a},\mathsf{t}_0)$ *is delivered by the datastream, then* $\mathsf{S}(\mathsf{a},t)$ *is true for every* $t \geq \mathsf{t}_0$.

*Thus,* $\langle [X := \mathsf{a}], \{\mathsf{R}(\mathsf{a}, T-k)\}\rangle \in \mathcal{H}(\langle \mathsf{S}(X,T), \Pi''\rangle, D, 0)$ *for all* $k$*. The pre-processing step needs to output this infinite set, so it cannot terminate.* ◁

We establish termination of the pre-processing step for two different classes of queries. A query $Q = \langle P, \Pi\rangle$ is *connected* if each rule in $\Pi$ contains at most one temporal variable, which occurs in the head whenever it occurs in the body; and it is *nonrecursive* if the directed graph induced by its dependencies is acyclic, cf. [Ronca et al. 2018b].

Proposition 10. *Let* $Q = \langle P, \Pi\rangle$ *be a nonrecursive and connected query. Then the set of all computed answers with premises to* $Q$ *over* $D_{-1}$ *can be computed in finite time.*

Proof. Let $T$ be the (only) temporal variable in $P$. We first show that all SLD-derivations for $\Pi \cup \neg\{P\}$ have a maximum depth. First, associate to each predicate $R$ the maximum length of a path in the dependency graph for $\Pi$ starting from $R$. Next, associate to each goal the sorted sequence of these values for each of its atoms. Then each resolution step decreases the sequence assigned to the goal with respect to the lexicographic ordering. Since this ordering is well-founded, the SLD-derivation must terminate.

Furthermore, since $\Pi$ is finite, there is a finite number of possible descendants for each node. Therefore, the tree containing all possible SLD-derivations for $\Pi \cup \{\neg P\}$ is a finite branching tree with finite height, and by König's Lemma it is finite.

Since each resolution step terminates (possibly with failure) in finite time, this tree can be built in finite time. □

The number of leaves in this tree can in theory be extremely high. Consider a program $\Pi$ whose dependency graph contains only paths of length at most $n$. Let $r$ be the maximum number of rules

defining a predicate in $\Pi$ and $b$ be the maximum number of literals in the body of a rule in $\Pi$. We give a rough estimate of the number of leaves in the corresponding SLD-tree by considering a computation rule that always chooses the oldest atom in the goal. The initial goal can then generate a maximum of $r$ descendants, each of which may contain up to $b$ literals. The next $b$ iterations will go through these literals, generating (for each node) a maximum of $r^b$ new nodes with up to $b^2$ literals each. This process can be repeated at most $n$ times, giving us a maximum number of leaves of

$$\prod_{i=0}^{n} r^{b^i} = r^{\frac{b^{n+1}-1}{b-1}}$$

Of course, this is a very rough over-approximation. Nevertheless, it is easy to construct a program $\Pi$ whose full SLD-tree is exponentially large in the number of predicate symbols in $\Pi$ (for example, if every intensional atom unifies with the head of two different rules).

The algorithm implicit in the proof of Proposition 10 can be improved by standard techniques (e.g. by keeping track of generated nodes to avoid duplicates). However, since it is a pre-processing step that is done offline and only once, we do not discuss such optimizations.

The authors of [Ronca et al. 2018a] also formally define query delay[3] as follows, where $Q$ is a query with temporal variable $T$.

Definition 9. *A* delay *for $Q$ is a natural number $d$ such that: for every substitution $\theta$ and every $\tau \geq T\theta + d$, $\theta \in \mathcal{A}(Q, D, \tau)$ iff $\theta \in \mathcal{A}(Q, D, T\theta + d)$.*

We show that the pre-processing step terminates when a delay exists. The proof is in two steps; we first show a lemma allowing us to ignore some types of nodes.

Lemma 11. *If $Q = \langle P, \Pi \rangle$ has delay $d$, then there exist natural numbers $d'$ and $d''$ such that: if an atom in a goal in an SLD-derivation for $\Pi \cup \{\neg P\}$ contains a temporal argument $T + k$ with $k > d'$ or $k < -d''$, then the SLD-tree with that atom as root does not contain a leaf including an extensional atom with a temporal parameter dependent on $T$.*

Proof. Assume that the thesis does not hold, i.e. for any $d'$ and $d''$ there exists an atom in an SLD-derivation for $\Pi \cup \{\neg P\}$ with temporal argument $T + k$ such that $k > d'$ or $k < -d''$ such that the SLD-tree with that atom as root contains a leaf including an extensional atom with temporal parameter dependent on $T$. By the independence of the computation rule, this property must hold for any SLD-derivation for $\Pi \cup \{\neg P\}$.

Then there is a predicate symbol $p$ that occurs infinitely many times in the tree for $\Pi \cup \{\neg P\}$ with distinct instantiations of the temporal argument. Since the number of distinct partial instantiations for the non-temporal arguments is finite (up to $\alpha$-equivalence), there must be at least one instantiation that occurs infinitely often, i.e. the tree contains distinct nodes with atoms $p(t_1, \ldots, t_n, T + k_1), \ldots, p(t_1, \ldots, t_n, T + k_m), \ldots$ where $k_1, \ldots, k_m, \ldots$ are all distinct.

Suppose that the SLD-tree that has $\leftarrow p(t_1, \ldots, t_n, T + k_i)$ as root contains a leaf with an extensional atom whose temporal argument depends on $T$, say $q(t'_1, \ldots, t'_j, T + \ell_i)$. The only way in which this can happen is if the temporal argument $T + k_i$ is never unified with a constant. This means that, for each $m$, the SLD-tree for $p(t_1, \ldots, t_n, T + k_m)$ must also contain a leaf including $q(t'_1, \ldots, t'_j, T + \ell_m)$, where $\ell_m - k_m = \ell_i - k_i$. Since the derivations for different atoms in a goal do not interfere, this means that all these atoms appear in different leaves of the initial SLD-tree.

For any $m$, we can construct a data stream $D$ and substitution $\theta$ such that $D$ consists of the extensional atoms in a leaf containing $q(t'_1, \ldots, t'_j, T\theta + \ell_m)$. Then $\theta \notin \mathcal{A}(Q, D, T\theta + t)$ for any $t < \ell_m$,

[3]These are not communication delays, but delays in producing answers to queries.

but clearly $\theta \in \mathcal{A}(Q, D, T\theta + t)$ for some $t \geq \ell_m$, so no number smaller than $\ell_m$ can be a delay for $Q$. □

Proposition 12. *If $Q$ has a delay $d$, then the set of all computed answers with premises to $Q$ over $D_{-1}$ can be computed in finite time.*

Proof. Let $d'$ and $d''$ be the two natural numbers guaranteed to exist by the previous lemma. Then, if a goal contains a positive literal with temporal parameter $T + k$ with $k > d'$ or $k < -d''$, we know that the SLD-tree with that atom as root cannot include leaves with temporal parameter depending on $T$. Hence, such a subtree either contains only failed branches, or it contains leaves that do not depend on $T$ (and therefore are always the same), or it is infinite.

Although we do not know the values of $d'$ and $d''$, we can still use this knowledge by performing the construction of the tree in a depth-first fashion. In order to do this, we choose a computation rule that delays atoms with temporal variable other than $T$. For each node with temporal variable $T$, before expanding it we first check whether the atom we are processing has already appeared before (modulo temporal variables), and if so we skip the construction of the corresponding subtree by directly inserting the corresponding leaves.

If an atom contains a temporal variable other than $T$, all leaves in its subtree contain no extensional predicates with non-constant temporal argument (otherwise there would not be a delay for $T$). Therefore these sub-goals can also be treated uniformly.

Finally, suppose that a branch is infinite. Again using a finiteness argument, any of its branches must necessarily at some point include repeated literals. In this case we can immediately mark it as failed. □

In general, given a query $Q = \langle P, \Pi \rangle$, we can try to compute a finite set $\mathcal{P}_Q$ that represents $\mathcal{H}(Q, D, -1)$ as follows. As in the proof of the previous proposition, choose a computation rule that delays atoms with temporal variable other than $T$. Build an SLD-tree with $\neg P$ as initial goal, with the following two extra checks.

- Before expanding a node, check whether the atom being processed has already appeared before (modulo temporal variables), and if so immediately generate the corresponding leaves.
- If the chosen atom has already appeared in the derivation, immediately mark the branch as failed.

Definition 10. *$\mathcal{P}_Q$ is the set of all entries $\langle \theta, \{\alpha_i\}_i \rangle$ such that the SLD-tree for $\neg P$ contains a leaf with computed answer $\langle \theta, \wedge_i \alpha_i \rangle$ with premises to $Q$ over $D_{-1}$ such that $\{\alpha_i\}_i$ is minimal in the whole tree.*

Each tuple $\langle \theta, H \rangle \in \mathcal{P}_Q$ represents the set of all hypothetical answers $\langle \theta\sigma, H\sigma \rangle$ as in Theorem 1.

*Remark.* The hypotheses in Propositions 10 and 12 are not necessary to guarantee termination of the algorithm presented for the pre-processing step. Indeed, consider the following example.

Example 8. *In the context of our running example, we say that a turbine has a manufacturing defect if it exhibits two specific failures during its lifetime: at some time it overheats, and at some (different) time it does not send a temperature reading.*

*Since this is a manufacturing defect, it holds at timepoint 0, regardless of when the failures actually occur.*[4] *We can model this property by the rule*

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

[4]This is actually an example of a rigid predicate, which we model as described.

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

The program in this example includes a rule that uses two different time variables. As a consequence, the query is not connected, and it does not have a delay (since no single predicate can use both $T_1$ and $T_2$). Still, the pre-processing step terminates.

We assume from this point onwards that the query $Q$ is connected.

### 5.3 The online step

We now describe how to compute and update the set $\mathcal{E}(Q,D,\tau)$ in a schematic way, i.e., using variables to represent families of similar hypotheses.

Definition 11. *A* schematic supported answer *for a query $Q$ at time $\tau$ is a triple $\langle \theta, E, H \rangle$ where $\theta$ is a substitution, $E$ is a set of ground EDB atoms and $H$ is a set of (not necessarily ground) EDB atoms such that $\langle \theta, E, H\sigma \rangle \in \mathcal{E}(Q,D,\tau)$ for every ground substitution $\sigma$.*

Definition 12. *A substitution $\sigma$ is a* minimal substitution with property $P$ *if: for any substitution $\theta$ with property $P$, there exists $\rho$ such that $\theta = \sigma\rho$.*

In particular, an mgu for a set is a minimal substitution unifying all formulas in the set.

Definition 13. *Let $Q$ be a query. For each time point $\tau$, we define the set $\mathcal{S}_\tau$ as follows.*

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

In § 5.4, we prove that $\mathcal{S}_\tau$ is a set of schematic supported answers that captures the set $\mathcal{E}(Q,D,\tau)$ precisely. Before that, we show that $\mathcal{S}_\tau$ can be computed as follows.

Definition 14. *Let $S$ be a set of schematic supported answers for a query $Q$ at time $\tau - 1$ and $\mathcal{P}_Q$ be the result of pre-processing $Q$. The* update *of $S$ using $\mathcal{P}_Q$, $\mathcal{U}(S, \mathcal{P}_Q)$, is computed as follows.*

- *For each $\langle \theta, H \rangle \in \mathcal{P}_Q$, let $M \subseteq H$ be the set of atoms with minimal timestamp. For each computed answer $\sigma$ to $\neg \bigwedge M$ and $D|_\tau$, add $\langle \theta\sigma, M\sigma, (H \setminus M)\sigma \rangle$ to $\mathcal{U}(S, \mathcal{P}_Q)$.*
- *For each $\langle \theta, E, H \rangle \in S$, let $M \subseteq H$ be the set of atoms with timestamp $\tau$. For each computed answer $\sigma$ to $\neg \bigwedge M$ and $D|_\tau$, add $\langle \theta\sigma, (E \cup M)\sigma, (H \setminus M)\sigma \rangle$ to $\mathcal{U}(S, \mathcal{P}_Q)$.*

Lemma 13. *For all $\tau \geq 0$, $\mathcal{S}_\tau = \mathcal{U}(\mathcal{S}_{\tau-1}, \mathcal{P}_Q)$.*

Proof. Both cases of the computation of $\mathcal{U}$ are similar. We show the first one in detail. Assume that $\langle \theta, H \rangle \in \mathcal{P}_Q$ and $\sigma$ is a minimal substitution such that $H\sigma \cap D|_\tau \neq \emptyset$ and $H\sigma \setminus D|_\tau$ only contains potentially future atoms wrt $\tau$. By definition of potentially future atom and the fact that all the elements of $D|_\tau$ have the same timestamp ($\tau$), the set $H\sigma \cap D|_\tau$ contains the elements of $H\sigma$ with minimal timestamp. Therefore $\sigma$ is a minimal substitution unifying all the elements of $M$ with elements of $D|_\tau$, and the thesis follows by soundness and completeness of SLD-resolution. Note that $M$ cannot be empty, since $H \neq \emptyset$.

For the second case, we observe that all elements of $H$ have their timestamp instantiated (by connectedness, there can be at most one temporal variable, which is instantiated when new elements are added from $\mathcal{P}_Q$). The proof then proceeds as in the previous case. □

Example 9. *We illustrate this mechanism with our running example, where*

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

*The atom with minimal timestamp in $H$ is* $\mathsf{Temp}(X, \mathsf{high}, T)$. *We start by setting* $\mathcal{S}_{-1} = \emptyset$.

*Since* $D|_0 = \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0)\}$, *SLD-resolution between* $\mathsf{Temp}(X, \mathsf{high}, T)$ *and* $D|_0$ *yields* $[X := \mathsf{wt25}, T := 0]$. *Therefore,*

$$\mathcal{S}_0 = \langle [X := \mathsf{wt25}, T := 0]\,, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0)\}, \underbrace{\{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}}_{H_0}\rangle\}\,.$$

*Next,* $D|_1 = \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1)\}$. *As before, SLD-resolution between* $\mathsf{Temp}(X, \mathsf{high}, T)$ *and* $D|_1$ *yields* $[X := \mathsf{wt25}, T := 1]$. *Furthermore, SLD-resolution between* $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1)$ *and* $D|_1$ *yields* $\emptyset$.

*Therefore,*

$$\begin{aligned}\mathcal{S}_1 = \{&\langle [X := \mathsf{wt25}, T := 1]\,, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1)\}, \underbrace{\{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 3)\}}_{H_1'}\rangle,\\ &\langle [X := \mathsf{wt25}, T := 0]\,, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1)\}, \underbrace{\{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}}_{H_1}\rangle\}\,.\end{aligned}$$

*Next we have* $D|_2 = \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}$. *Again, SLD-resolution between* $\mathsf{Temp}(X, \mathsf{high}, T)$ *and* $D|_2$ *yields the substitution* $[X := \mathsf{wt25}, T := 2]$, *while SLD-resolution between* $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)$ *(the atom with minimal timestamp in both* $H_1$ *and* $H_1'$*) and* $D_3$ *yields* $\emptyset$. *Thus*

$$\begin{aligned}\mathcal{S}_2 = \{&\langle [X := \mathsf{wt25}, T := 2]\,, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}, \underbrace{\{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 3), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 4)\}}_{H_2''}\rangle,\\ &\langle [X := \mathsf{wt25}, T := 1]\,, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}, \underbrace{\{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 3)\}}_{H_2'}\rangle,\\ &\langle [X := \mathsf{wt25}, T := 0]\,, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}, \emptyset\rangle\}\,.\end{aligned}$$

*The last element of* $\mathcal{S}_2$ *has an empty set of hypotheses, so the answer* $[X := \mathsf{wt25}, T := 0]$ *is known at this point.*

*Suppose now that* $D|_3 = \emptyset$. *Then SLD-resolution between* $\mathsf{Temp}(X, \mathsf{high}, T)$ *and* $D|_3$ *fails, so no new elements are added from* $\mathcal{P}_Q$ *to* $\mathcal{S}_3$. *Furthermore,* $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 3)$ *(which appears in both* $H_2'$ *and* $H_2''$*) does not unify with* $D|_3$, *and we know that it cannot be delivered by the data stream. Therefore*

$$\mathcal{S}_3 = \langle [X := \mathsf{wt25}, T := 0]\,, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}, \emptyset\rangle\}\,.$$

◁

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

### 5.4 Correctness

We now show that the algorithm given in the previous section indeed maintains a set of schematic supported answers to $Q$ that captures all supported answers at any given time point.

Theorem 3 (Soundness). *If $\langle \theta, E, H\rangle \in \mathcal{S}_\tau$, $E \neq \emptyset$, and $\sigma$ instantiates all free variables in $E \cup H$, then $\langle \theta\sigma, H', E\sigma\rangle \in \mathcal{E}(Q, D, \tau)$ for some $H' \subseteq H\sigma$.*

Proof. First, we show by induction on $\tau$ that there is an SLD-derivation with future premises of $Q$ and $D_\tau$ with computed answer with premises $\langle \theta, H\rangle$. For $\mathcal{S}_{-1} = \emptyset$ this is trivially the case.

Suppose now that $\tau \geq 0$ and $\langle \theta, E, H\rangle \in \mathcal{S}_\tau$. By Lemma 13, $\langle \theta, E, H\rangle$ can be computed by SLD-resolution from $M \subseteq H'$ for either some $\langle \theta', H'\rangle \in \mathcal{P}_Q$ or some $\langle \theta', E', H'\rangle \in \mathcal{S}_{\tau-1}$. In both cases, we can compose this SLD-derivation with the one obtained either by the way $\mathcal{P}_Q$ is constructed or the one obtained by induction hypothesis to obtain an SLD-derivation with future premises of $Q$ and $D_\tau$ – this simply requires adding the remaining elements of $H'$ to all nodes in the derivation from Lemma 13.

By applying Theorem 1 to this SLD-derivation, we conclude that $\langle \theta\sigma, H'\rangle \in \mathcal{H}(Q, D, \tau)$ for some $H' \subseteq H\sigma$. By construction, $E\sigma \neq \emptyset$ is evidence for this answer. □

It may be the case that $\mathcal{S}_\tau$ contains some elements that do not correspond to hypothetical answers because of the minimality requirement. Consider a simple case of a query $Q$ where $\mathcal{P}_Q = \{\langle \emptyset, \{p(a,0), p(b,1), p(c,2)\}\rangle, \langle \emptyset, \{p(a,0), p(c,2), p(d,3)\}\rangle\}$, $D|_0 = \{p(a,0)\}$ and $D|_1 = \{p(b,1)\}$. Then $\mathcal{S}_1 = \{\langle \emptyset, \{p(a,0), p(b,1)\}, \{p(c,2)\}\rangle, \langle \emptyset, \{p(a,0)\}, \{p(c,2), p(d,3)\}\rangle\}$, and the second element of this set has a non-minimal set of hypotheses. We chose not to include a test for set inclusion in the definition of $\mathcal{S}_\tau$, though, for efficiency reasons.

Theorem 4 (Completeness). *If $\langle \sigma, H, E\rangle \in \mathcal{E}(Q, D, \tau)$, then there exist a substitution $\rho$ and a triple $\langle \theta, E', H'\rangle \in \mathcal{S}_\tau$ such that $\sigma = \theta\rho$, $H = H'\rho$ and $E = E'\rho$.*

Proof. By Theorem 2, $\langle Q, D_\tau\rangle \vdash_{\mathsf{SLD}} \langle \theta, \wedge_i \alpha_i\rangle$ for some substitution $\rho$ and set of atoms $H' = \{\alpha_i\}_i$ with $H = \{\alpha_i\rho\}_i$ and $\sigma = \theta\rho$ for some $\theta$. By Proposition 9, there is a stratified SLD-derivation computing this answer. The sets of atoms from $D|_\tau$ unified in each stratum of this derivation define the set of elements from $H$ that need to be unified to construct the corresponding element of $\mathcal{S}_\tau$, and it is straightforward to check that they are defined as in Lemma 13. □

It also follows from our construction that: if $d$ is a query delay for $Q$, then the timestamp of each element of $H$ is at most $\tau + d$. Likewise, if $w$ is a window size for $Q$, then all elements in $E$ must have timestamp at least $\tau - w$.

The following example also shows how, by outputting hypothetical answers, we can answer queries earlier than in other formalisms.

Example 10. *Suppose that we extend the program $\Pi_E$ in our running example with the following rule (as in Example 2 from [Ronca et al. 2018b]).*

$$\mathsf{Temp}(X, \mathsf{n/a}, T) \rightarrow \mathsf{Malf}(X, T)$$

*If $D_1 = \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0), \mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 1), \mathsf{Temp}(\mathsf{wt42}, \mathsf{n/a}, 1)\}$, then*

$$\langle [T := 0, X := \mathsf{wt25}], \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, i) \mid i = 0, 1\}, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)\}\rangle,$$
$$\langle [T := 1, X := \mathsf{wt42}], \{\mathsf{Temp}(\mathsf{wt42}, \mathsf{n/a}, 1)\}, \emptyset\rangle$$

*are both in $\mathcal{S}_1$. Thus, the answer $[T := 1, X := \mathsf{wt42}]$ is produced at timepoint 1, rather than being delayed until it is known whether $[T := 0, X := \mathsf{wt25}]$ is an answer.* ◁

Theorem 5 (Complexity). *The set $\mathcal{S}_\tau$ can be computed from $\mathcal{P}_Q$ and $\mathcal{S}_{\tau-1}$ in time polynomial in the size of $\mathcal{P}_Q$, $\mathcal{S}_{\tau-1}$ and $D|_\tau$.*

Proof. If $\langle \theta, H\rangle \in \mathcal{P}_Q$, then we can compute the set $M$ of elements of $H$ with minimal timestamp in linear time. To decide which substitutions make $M$ a subset of $D|_\tau$, we can perform classical

SLD-resolution between $M$ and $D|_\tau$. For each such element of $\mathcal{P}_Q$, the size of every SLD-derivation that needs to be constructed is bound by the number of atoms in the initial goal, since $D|_\tau$ only contains facts. Furthermore, all unifiers can be constructed in time linear in the size of the formulas involved, since the only function symbol available is addition of temporal terms. Finally, the total number of SLD-derivations that needs to be considered is bound by the size of $\mathcal{P}_Q \times D|_\tau$.

If $\langle \theta, E, H \rangle \in \mathcal{S}_{\tau-1}$ and $E \neq \emptyset$, then again we can compute in linear time the set of facts in $H$ that must unify with $D|_\tau$ – these are the elements of $H$ whose timestamp is exactly $\tau$. As above, the elements that must be added to $\mathcal{S}_\tau$ can then be computed in polynomial time by SLD-resolution. □

It would be interesting to generalize our algorithm to non-connected queries, in order to deal with examples such as Example 8. There, if $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0) \in D_0$, it would be nice to obtain

$$\langle [X := \mathsf{wt25}, T_1 := 0], \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 0)\}, \{\mathsf{Temp}(\mathsf{wt25}, \mathsf{n/a}, T_2)\}\rangle \in \mathcal{S}_0$$

as this can be relevant information, even if we do not know when (or whether) it will lead to an answer to the original query. However, such a generalization requires substantial changes to our approach, and we leave it for future work.

## 6 ADDING NEGATION

We now show how we can extend our framework to include negation. We extend the syntax of our programs to allow negated atoms in the bodies of rules, but not in heads of rules or in queries. The semantics of negation is based on the closed world assumption.

Hypothetical answers need to deal with the fact that this semantics introduces non-monotonicity: if $\neg p$ holds at a given point in time, $p$ may still become true later due to new incoming data. Therefore $\neg p$ can only be taken as evidence for a hypothetical answer when the current $\tau$-history allows us to prove that $p$ does not hold in the whole dataset. We deal with this issue by giving supported answers a semantics reminiscent of the Kripke semantics for intuitionistic logic.

### 6.1 Temporal Datalog with negation

To the best of our knowledge, the existing works on Temporal Datalog only consider the negation-free fragment of the language. We extend the syntax of Temporal Datalog in the standard way [Abiteboul et al. 1995], by allowing bodies of rules to contain negated atoms.

More specifically, the syntax of the language is defined exactly as for Temporal Datalog (Section 2.1), with the following two changes:

- a rule now has the form $\wedge_i \ell_i \rightarrow \alpha$, where each $\ell_i$ is a temporal literal and $\alpha$ is a temporal atom;
- we require negation to be safe, i.e. if a rule $r$ contains a negative literal $\neg p(t_1, \ldots, t_n)$, then all variables appearing in $t_1, \ldots, t_n$ also occur in a positive literal in $r$ (possibly the head).

We illustrate this extended language with the example that we use throughout this section.

Example 11. *A hospital is conducting a study on early detection of critical patients. A patient that either shows bad results on their lab tests or does not have good vital signs (cardiac markers and blood oxygen levels) is placed under observation. A patient that is under observation and does not have good vital signs is moved to the ICU. This patient is considered a candidate for early risk detection two days before being moved to the ICU.*

*These rules are implemented in the following program* $\Pi_H$*, which uses EDB predicates* BCA *(bad clinical analyses),* GCM *(good cardiac markers) and* GBOL *(good blood oxygen levels), as well as IDB predicates* GVS *(good vital signs),* ST *(patient status) and* Risk *(patient at risk). All these predicates include the patient's name as the first argument and the timestamp as the last; predicate* ST *also includes the patient status (*us *for "under surveillance" and* ic *for "intensive care") as second argument.*

*Since the laboratory is overworked, they only communicate bad results of analyses as a combined outcome (predicate* BCA*) after two days.*

$$\begin{aligned}
\mathsf{GCM}(X,T), \mathsf{GBOL}(X,T) &\longrightarrow \mathsf{GVS}(X,T)\\
\mathsf{BCA}(X,T+2) &\longrightarrow \mathsf{ST}(X,\mathsf{us},T+1)\\
\neg\mathsf{GVS}(X,T), \neg\mathsf{ST}(X,\mathsf{us},T) &\longrightarrow \mathsf{ST}(X,\mathsf{us},T+1)\\
\neg\mathsf{GVS}(X,T), \mathsf{ST}(X,\mathsf{us},T) &\longrightarrow \mathsf{ST}(X,\mathsf{ic},T+1)\\
\mathsf{ST}(X,\mathsf{ic},T+2) &\longrightarrow \mathsf{Risk}(X,T)
\end{aligned}$$

*Note that* $\mathsf{BCA}(X,T+2)$ *actually conveys information about the patient's status at timestamp* $T$*. (In particular,* $\mathsf{BCA}(X,0)$ *and* $\mathsf{BCA}(X,1)$ *are never produced by the datastream for any* $X$*.)* ◁

If $\Pi$ is a program and $D = \{D|_\tau \mid \tau \in \mathbb{N}\}$ is a dataset, we write $\Pi \cup D \models \ell$ to denote that $\ell$ is entailed by $\Pi$ and $\bigcup_{\tau\in\mathbb{N}} D|_\tau$ according to the cautious answer set semantics for logic programming [Gelfond and Lifschitz 1988]. As usual, a precise correspondence between this semantics and an operational semantics based on SLD-resolution only holds in the presence of a stratification of the predicate symbols in the program with respect to negative dependencies [Lloyd 1984]. We now discuss what a notion of stratification should look like for programs in Temporal Datalog with negation.

Definition 15. *The* temporal closure *of a program* $\Pi$ *in Temporal Datalog with negation is the program* $\Pi^\downarrow$ *defined as follows. For each* $(n+1)$*-ary predicate symbol* $p$ *with a temporal argument in the signature underlying* $\Pi$*, the signature for* $\Pi^\downarrow$ *contains a family of* $n$*-ary predicate symbols* $\{p_t\}_{t\in\mathbb{N}}$*. For each rule in* $\Pi$*,* $\Pi^\downarrow$ *contains all rules obtained by instantiating its temporal parameter in all possible ways and replacing* $p(x_1,\ldots,x_n,t)$ *by* $p_t(x_1,\ldots,x_n)$*.*

Observe that $\Pi^\downarrow$ is an infinite program (in Datalog with negation) as long as $\Pi$ has at least one predicate with a temporal argument.

Recall that, for infinite programs, a stratification of $\Pi$ 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 $\Pi_k$, (ii) if $p$ occurs in the body of a rule with head $q$, then the definition of $q$ is contained in $\Pi_i$ for some $i$ such that $i \geq k$ (if the occurrence is not negated) or $i > k$ (if the occurrence is negated) [Kolaitis 1991].

Definition 16. *A program* $\Pi$ *is* $T$*-stratified if* $\Pi^\downarrow$ *is stratified.*

This definition of stratification is reminiscent of the concepts of locally stratified [Przymusinski 1988a,b] and temporally stratified [Zaniolo 2015]. However, it differs from both. Local stratification has mostly been studied in the context of programs with a finite number of constants, which is not our case (there are infinitely many time points). On the other hand, temporal stratification as in [Zaniolo 2015] requires each stratum to contain exactly the predicates with the same time parameter (i.e., $\Pi_i$ contains all rules whose head uses predicate symbols indexed by $i$). We make no such assumption in this work.

Example 12. *Program* $\Pi_H$ *is* $T$*-stratified with the following strata:*

$$\begin{aligned}
\Pi_0 &= \{\mathsf{GCM}_t, \mathsf{GBOL}_t, \mathsf{GVS}_t, \mathsf{BCA}_t \mid t \in \mathbb{N}\} \cup \{\mathsf{ST}_0\}\\
\Pi_1 &= \{\mathsf{ST}_1\}\\
\Pi_{t+2} &= \{\mathsf{ST}_{t+2}, \mathsf{Risk}_t\} \qquad t \geq 0
\end{aligned}$$ ◁

Given an interpretation $I$, satisfaction of rules and programs is defined as before. (Note that the restriction that negation be safe removes any potential ambiguity.) It is easy to check that $I$ is a

model of $\Pi$ iff it is a model of $\Pi^{\downarrow}$, and as a consequence $T$-stratified programs have a unique answer set that coincides with their well-founded model.

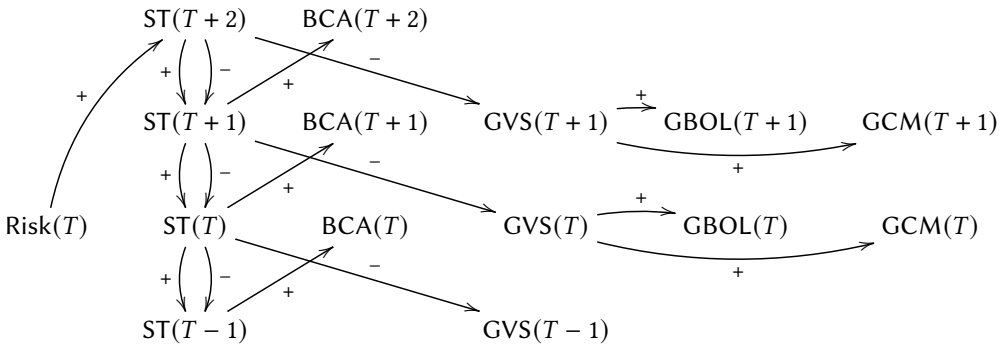
For finite programs, stratification can be phrased (and decided) as a property of the graph of dependencies between the predicate symbols in the program. In our case, such a graph is infinite, and such a procedure may not terminate. However, the usual construction can be adapted to our framework in order to provide a decision procedure.

Definition 17. *Let $\Pi$ be a program. The* temporal dependency graph *of $\Pi$, $\mathcal{G}_\Pi$, is the smallest graph satisfying the following properties:*

*(1) for each predicate symbol $p$ with a temporal argument, $\mathcal{G}_\Pi$ includes a node with label $p(T)$ (where $T$ is a formal temporal variable);*
*(2) for each predicate symbol $p$ without temporal arguments, $\mathcal{G}_\Pi$ includes a node with label $p$;*
*(3) for each node with label $p(T+k)$ with $k \geq 0$ and each rule $r$ in the definition of $p$, let $\theta$ be the substitution that makes the temporal variable in the head of $r$ equal to $T+k$:*
  *(a) for each positive literal $q(x_1, \ldots, x_n, T+k')$ in $\mathsf{body}(r\theta)$, $\mathcal{G}_\Pi$ includes an edge between $p(T+k)$ and $q(T+k')$ with label $+$.*
  *(b) for each negative literal $\neg q(x_1, \ldots, x_n, T+k')$ in $\mathsf{body}(r\theta)$, $\mathcal{G}_\Pi$ includes an edge between $p(T+k)$ and $q(T+k')$ with label $-$.*

The graph $\mathcal{G}_\Pi$ can be constructed by starting with the nodes described in the first two conditions, adding the edges (and new nodes) required by conditions 3(a) and 3(b), and iterating the process for any newly created nodes.

Example 13. *The temporal dependency graph for the program $\Pi_H$ is depicted below.*




◁

Proposition 14. *There is an algorithm that decides whether a program $\Pi$ is $T$-stratified.*

Proof. We first show that we can decide whether $\mathcal{G}_\Pi$ contains a path that passes infinitely many times through distinct edges labeled with $-$. Indeed, we can build a finite subgraph $\mathcal{G}^-_\Pi$ of $\mathcal{G}_\Pi$ by not expanding any nodes labeled $p(T+k)$ if $\mathcal{G}^-_\Pi$ already contains a path from $p(T+k')$ to $p(T+k)$ for some $k' < k$. Since $\Pi$ only contains a finite number of rules with a finite number of literals in their bodies, this condition generates an upper bound on the timestamps of the nodes that appear in the graph; since the number of predicate symbols is finite, $\mathcal{G}^-_\Pi$ must also be finite. Furthermore, $\mathcal{G}_\Pi$ contains a path in the conditions described if either (i) $\mathcal{G}^-_\Pi$ also contains such a path (i.e. it contains a loop including an edge labeled with $-$) or (ii) $\mathcal{G}^-_\Pi$ contains a path from $p(T+k')$ to $p(T+k)$ with $k' < k$ that includes an edge labeled with $-$.

Now we show that, if $\mathcal{G}_\Pi$ does not contain a path that passes infinitely many times through edges labeled with $-$, then $\Pi$ is $T$-stratified.

Let $\mathcal{G}_{\Pi^\downarrow}$ be the graph of dependencies for $\Pi^\downarrow$. For each predicate symbol $p_t$ in $\Pi^\downarrow$, the graph $\mathcal{G}_{\Pi^\downarrow}$ contains a subgraph isomorphic to a subgraph[5] of the connected component of $\mathcal{G}_\Pi$ containing $p(T)$. In particular, $\mathcal{G}_{\Pi^\downarrow}$ contains infinitely many copies of $\mathcal{G}_\Pi$.

Suppose that $\mathcal{G}_{\Pi^\downarrow}$ contains a path that passes infinitely many times through edges labeled with $-$, and let $p_t$ be a node in it with minimum value of $t$ (if $t$ occurs multiple times in the path, choose any one of its occurrences). Consider the isomorphic copy of $\mathcal{G}_\Pi$ embedded in $\mathcal{G}_{\Pi^\downarrow}$ where $p(T)$ is mapped to $p_t$. The whole path starting from $p_t$ must be contained in this copy, since the timestamps of all corresponding nodes in $\mathcal{G}_\Pi$ are greater or equal to $T$.

Using these two properties, we can construct a stratification of $\Pi^\downarrow$ in the usual inductive way.

- $\mathcal{G}_0 = \mathcal{G}_{\Pi^\downarrow}$
- For each $i \geq 0$, $\Pi_i$ contains all predicates labeling nodes $n$ in $\mathcal{G}_i$ such that: all paths from $n$ contain only edges labeled with $+$.
- For each $i \geq 0$, $\mathcal{G}_{i+1}$ is obtained from $\mathcal{G}_i$ by removing all nodes with labels in $\Pi_i$ and all edges to those nodes.

The only non-trivial part of showing that $\Pi_0, \ldots, \Pi_n, \ldots$ is a stratification of $\Pi^\downarrow$ is guaranteeing that every predicate symbol $p_t$ is contained in $\Pi_k$ for some $k$. This is established by induction: if any path from $p_t$ contains at most $n$ edges labeled with $-$, then $p_t \in \Pi_n$. We now show that, for any $p_t$, there is such a bound on the number of edges labeled with $-$.

First observe that, in $\mathcal{G}_\Pi$, there is a bound $B$ on the number of edges labeled with $-$ in any path starting from $p(T)$: (i) if there is a path from $q(T + k_1)$ to $q(T + k_2)$ for some $q$ and $k_1 \leq k_2$ containing an edge labeled with $-$, then there would be a path with an infinite number of such edges, and (ii) since there is a finite number of predicate symbols in $\Pi$ we cannot build paths with an arbitrarily long number of edges labeled with $-$.

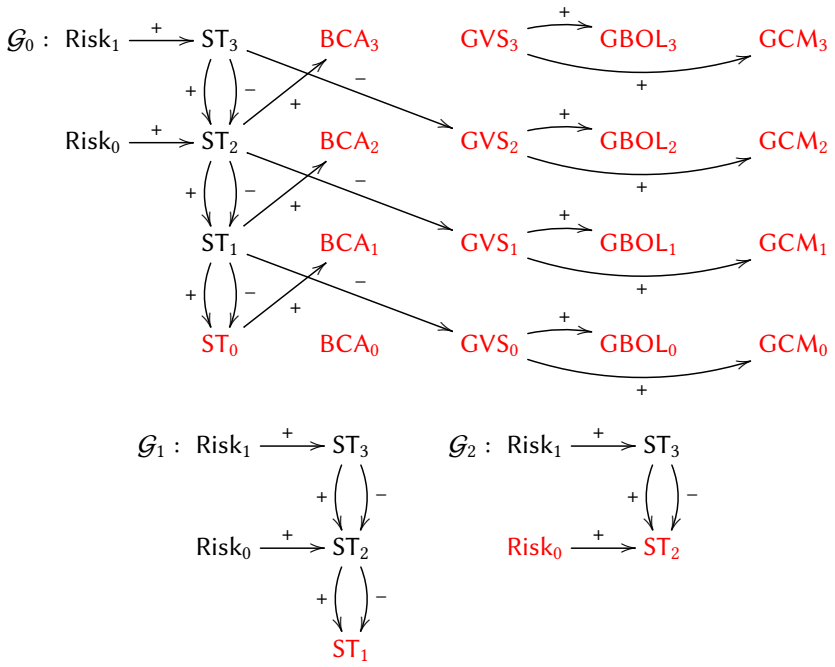
Now consider a path starting at $p_t$. If this path has length greater than $B$, then it cannot be contained completely in the subgraph of $\mathcal{G}_{\Pi^\downarrow}$ isomorphic to $\mathcal{G}_\Pi$ containing $p(T)$. Therefore, it must at some point reach a node $q_{t'}$ with $t' < t$. Likewise, the path starting at $q_{t'}$ corresponds to a path in a (different) subgraph of $\mathcal{G}_{\Pi^\downarrow}$ isomorphic to $\mathcal{G}_\Pi$, and as such cannot contain more than $B$ edges labeled with $-$. Since we can only decrease $t$ a finite number of times, the path cannot contain arbitrarily many edges labeled with $-$. □

Note that this does not contradict the classical undecidability of deciding whether a program has local stratification [Palopoli 1992], since our language does not have function symbols.

Example 14. *When the graph $\mathcal{G}_\Pi$ is finite, the proof of Proposition 14 can be used to construct any stratum $\Pi_n$ in finite time by constructing only the necessary part of $\mathcal{G}_{\Pi^\downarrow}$. In particular, applying it to $\mathcal{G}_{\Pi_H}$ shown earlier yields the stratification already presented in Example 12.*

[5]It may be a proper subgraph in case $t$ is low enough that some temporal arguments would become negative.

*For illustrative purposes, we show a fragment of the graphs $\mathcal{G}_0 = \mathcal{G}_{\Pi^\downarrow}$, $\mathcal{G}_1$ and $\mathcal{G}_2$, marking in color the nodes that are removed (i.e., collected in $\Pi_i$).*




◁

Since the temporal closure of a program in Temporal Datalog with negation is a normal program in logic programming (albeit with an infinite number of rules), the classical results from logic programming apply. In particular, we can define the well-founded model for a stratified program in Temporal Datalog with negation by transfinite induction, taking for each stratum the minimal model for the positive program obtained by (i) removing rules whose body contains a negative literal that is false in the model for the corresponding lower stratum and (ii) removing all negative literals in other rules. This is a model of the original program, and also its only answer set.

### 6.2 Hypothetical answers in the presence of negation

The inclusion of negation in the language poses a new problem with respect to defining the notion of evidence for a hypothetical answer: we need to be able to argue that some atom will never become true given the set of hypotheses. To formalize this notion, it is not enough to consider the data stream extended by $H$ – we need to account also for additional information that may make more atoms provable.

Definition 18. *Let $D = \{D|_\tau \mid \tau \in \mathbb{N}\}$ be a dataset, $\tau'$ be a time point and $H^+$ and $H^-$ be finite sets of ground EDB atoms. A dataset $D' = \{D'|_t \mid t \in \mathbb{N}\}$ is a* possible evolution *of $D$ compatible with $\langle H^+, H^-\rangle$ at time $\tau$ if:*

- *$D'|_\tau = D|_\tau$ for all $\tau \leq \tau'$;*
- *$H^+ \subseteq \bigcup_{\tau\in\mathbb{N}} D'|_t$;*
- *$H^- \cap (\bigcup_{\tau\in\mathbb{N}} D'|_t) = \emptyset$.*

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.

Hypothetical answers to a query $Q = \langle \Pi, P \rangle$ over $D_\tau$, where $\Pi$ may contain negated atoms, need to consider all possible evolutions of the dataset consistent with the hypotheses.

Given a set of literals $L$, we write $L^+$ for the set of positive literals in $L$ and $L^-$ for the set of atoms whose negation is in $L$. In particular, we use this notation in the next two definitions.

Definition 19. *A* hypothetical answer *to query $Q = \langle P, \Pi \rangle$ over $D_\tau$ is a pair $\langle \theta, H \rangle$, where $\theta$ is a substitution and $H$ is a finite set of ground EDB literals such that:*

- $\mathsf{supp}(\theta) = \mathsf{var}(P)$*;*
- $H^+$ *and* $H^-$ *only contain atoms future-possible for $\tau$;*
- $\Pi \cup (\bigcup_{\tau' \in \mathbb{N}} D'|_{\tau'}) \models P\theta$ *for each possible evolution $D'$ of $D$ compatible with $\langle H^+, H^- \rangle$ at time $\tau$;*
- $H$ *is minimal with respect to set inclusion.*

Definition 20. *Let $Q = \langle P, \Pi \rangle$ be a query, $D$ be a data stream and $\tau$ be a time instant. A set of ground EDB literals $E$ is* evidence *supporting $\langle \theta, H \rangle \in \mathcal{H}(Q, D, \tau)$ if:*

- $E^+ \subseteq D_\tau$ *and* $E^- \cap D_\tau = \emptyset$*;*
- $E^+ \cup E^- \neq \emptyset$*;*
- $\Pi \bigcup D' \models P\theta$ *for each possible evolution $D'$ of $D$ compatible with $\langle E^+ \cup H^+, E^- \cup H^- \rangle$ at time $-1$;*
- $E$ *is minimal with respect to set inclusion.*

*Note that the third condition effectively ignores $D$, since it ignores its contents at all time points. This matches our intuition that the evidence is exactly the information from the data stream that is relevant to prove the query.*

*A* supported answer *to $Q$ over $D_\tau$ is a triple $\langle \theta, H, E \rangle$ such that $\Pi \cup H \not\models P\theta$ and $E$ is evidence supporting $\langle \theta, H \rangle$.*

Intuitively, $E^+$ is the set of facts produced by the data stream that are essential to the hypothetical answer, while $E^-$ is the set of facts whose absence is relevant for the hypothetical answer.

We illustrate these alternative notions in the context of our running example.

Example 15. *Consider the program $\Pi_H$ from Example 11 and the query $Q_R = \langle \mathsf{Risk}(X, T), \Pi_H \rangle$, and let $\theta = [X := \mathsf{john}, T := 0]$.*

- *Suppose that $D|_0 = \emptyset$ then $\langle \theta, H \rangle$ where $H = \{\neg\mathsf{GCM}(\mathsf{john}, 1)\}$ is one (of several) hypothetical answers to $Q_R$ over $D_0$, with evidence $\{\neg\mathsf{GBOL}(\mathsf{john}, 0)\}$.*
  *Indeed* $\mathsf{GBOL}(\mathsf{john}, 0) \notin D|_0$*, so it can never hold; therefore* $\neg\mathsf{GVS}(\mathsf{john}, 0)$ *holds, and we can conclude* $\mathsf{ST}(\mathsf{john}, \mathsf{us}, 1)$*. If* $\mathsf{GCM}(\mathsf{john}, 1)$ *is never produced by the data stream, then* $\neg\mathsf{GVS}(\mathsf{john}, 1)$ *also holds, from which we can derive* $\mathsf{ST}(\mathsf{john}, \mathsf{ic}, 2)$ *and as a consequence* $\mathsf{Risk}(\mathsf{john}, 0)$*.*
- *Assume now that $D|_0 = \{\mathsf{GBOL}(\mathsf{john}, 0), \mathsf{GCM}(\mathsf{john}, 0)\}$. Now there is a hypothetical answer $\langle \theta, H' \rangle$ to $Q_R$ over $D_1$, where $H' = \{\mathsf{BCA}(\mathsf{john}, 2), \neg\mathsf{GCM}(\mathsf{john}, 1)\}$. Indeed,* $\mathsf{ST}(\mathsf{john}, \mathsf{us}, 1)$ *holds as long as* $\mathsf{BCA}(\mathsf{john}, 2)$ *is produced by the data stream; and if* $\mathsf{GCM}(\mathsf{john}, 1)$ *is not produced by the data stream, then as before* $\neg\mathsf{GVS}(\mathsf{john}, 1)$ *holds, from which we can derive* $\mathsf{ST}(\mathsf{john}, \mathsf{ic}, 2)$ *and* $\mathsf{Risk}(\mathsf{john}, 0)$*. At this stage, there is no evidence for $\langle \theta, H' \rangle$.* ◁

### 6.3 Pre-processing

The notion of computed answer with premises in the presence of negation is generalized by allowing leaves of SLD-resolution to contain negated (EDB or IDB) atoms besides positive EDB atoms.

Our two-stage algorithm can now be adapted to the language with negation using this extended notion. Let $Q = \langle P, \Pi \rangle$ be a query. In the pre-processing step, we compute the set $\mathcal{P}_Q$ using SLD-resolution as before. For each element $\langle \theta, \{\alpha_i\}_{i \in I} \rangle \in \mathcal{P}_Q$ and $i \in I$, if $\alpha_i$ is $\neg p(t_1, \ldots, t_n, t_{n+1})$, we

generate a new query $Q' = \langle p(t_1, \ldots, t_n, T), \Pi \rangle$, obtained by replacing the time parameter in $\alpha_i$ with a variable, and repeat the pre-processing step to compute $\mathcal{P}_{Q'}$. We iterate this construction until no fresh queries are generated.

Given that the number of constants is finite and the number of arguments in any predicate in $\Pi$ is finite, the total number of queries that can be generated is also finite – albeit quite high, as in the worst case we can generate $O(m \times c^{k-1})$ auxiliary queries, where $m$ is the number of predicate symbols in the language, $c$ is the number of constants, and $k$ is the highest arity of any predicate in $\Pi$. (The only possible source of unboundedness is the temporal parameter, which we keep uninstantiated.) Therefore, Propositions 10 and 12 still hold for this more expressive language, but there is an exponential increase in worst-case complexity of the pre-processing step.

Example 16. *We continue with our running example (Example 11). Applying SLD-resolution to* $\leftarrow \mathsf{Risk}(X, T)$ *until we reach a goal with only extensional or negated atoms results in two possible derivations:*

$$\leftarrow \mathsf{Risk}(X, T)$$
$$\downarrow$$
$$\leftarrow \mathsf{ST}(X, \mathsf{ic}, T + 2)$$
$$\downarrow$$
$$\leftarrow \neg\mathsf{GVS}(X, T + 1), \mathsf{ST}(X, \mathsf{us}, T + 1)$$
$$\swarrow \qquad\qquad \searrow$$
$$\leftarrow \neg\mathsf{GVS}(X, T + 1), \mathsf{BCA}(X, T + 2) \qquad\qquad \leftarrow \neg\mathsf{GVS}(X, T + 1), \neg\mathsf{GVS}(X, T), \neg\mathsf{ST}(X, \mathsf{us}, T)$$

*This yields the following set* $\mathcal{P}_{Q_R}$.

$$\mathcal{P}_{Q_R} = \{\langle \emptyset, \{\mathsf{BCA}(X, T + 2), \neg\mathsf{GVS}(X, T + 1)\}\rangle,$$
$$\langle \emptyset, \{\neg\mathsf{GVS}(X, T + 1), \neg\mathsf{GVS}(X, T), \neg\mathsf{ST}(X, \mathsf{us}, T)\}\rangle.$$

*Since* GVS *and* ST *appear as negated hypotheses, we generate two auxiliary queries* $Q_G = \langle \mathsf{GVS}(X, T), \Pi_H \rangle$ *and* $Q_S = \langle \mathsf{ST}(X, \mathsf{us}, T), \Pi_H \rangle$. *These queries are now pre-processed in turn.*

$$\mathcal{P}_{Q_G} = \langle \emptyset, \{\mathsf{GCM}(X, T), \mathsf{GBOL}(X, T)\}\rangle$$
$$\mathcal{P}_{Q_S} = \{\langle \emptyset, \{\mathsf{BCA}(X, T + 1)\}\rangle,$$
$$\langle \emptyset, \{\neg\mathsf{GVS}(X, T - 1), \neg\mathsf{ST}(X, \mathsf{us}, T - 1)\}\rangle$$

*Since the newly generated negated goals yield queries that have already been considered, the pre-processing step is finished.* ◁

*Constants in generated queries.* The negated atoms in leaves may be partly instantiated. This creates a need to decide whether those instantiations should be maintained, or whether all terms should be replaced by variables in the new query. The precise choice depends on the concrete instance of the problem: replacing all terms by variables restricts the total number of queries being managed, since all negated instances involving the same predicate reduce to the same query. However, having instantiated terms may substantially reduce the number of branches in the SLD-tree constructed, and the number of potential answers that have to be tracked. In particular, if the instance originates from a term that is instantiated in the original query, it might be counter-productive to replace it by a variable.

Example 17. *Consider the program* $\Pi$ *below and the query* $\langle \mathsf{Q}(T), \Pi \rangle$.

$$\neg\mathsf{P}(\mathsf{c}_1, T) \rightarrow \mathsf{Q}(T) \qquad \neg\mathsf{P}(\mathsf{c}_2, T) \rightarrow \mathsf{Q}(T)$$
$$\neg\mathsf{P}(\mathsf{c}_3, T) \rightarrow \mathsf{Q}(T) \qquad \mathsf{R}(X, T) \rightarrow \mathsf{P}(X, T)$$

*In this case, replacing the constants* $\mathsf{c}_1$, $\mathsf{c}_2$ *and* $\mathsf{c}_3$ *with a variable in the generated queries results in the same query* $\mathsf{P}(X,T)$ *in all cases. Keeping the constants in the generated queries results in the need to process* $\mathsf{P}(\mathsf{c}_1,T)$, $\mathsf{P}(\mathsf{c}_2,T)$ *and* $\mathsf{P}(\mathsf{c}_3,T)$*, which generates three isomorphic trees and nearly identical computed answers with premises.* ◁

Example 18. *Consider the following program* $\Pi$

$$\neg\mathsf{P}(X,T) \to \mathsf{Q}(X,T) \qquad\qquad \to \mathsf{P}(\mathsf{c},T)$$
$$\mathsf{R}(X,T) \to \mathsf{P}(X,T)$$

*and the query* $\langle \mathsf{Q}(\mathsf{c},T), \Pi\rangle$*. This query generates as premise* $\neg\mathsf{P}(\mathsf{c},T)$*, which can never hold because of the second rule. However, if* $\mathsf{c}$ *is abstracted to a variable, the generated query becomes* $\mathsf{P}(X,T)$*, which yields two computed answer with premises* $\langle [X := \mathsf{c}], \emptyset, \emptyset\rangle$ *and* $\langle \emptyset, \mathsf{R}(X,T), \emptyset\rangle$*.* ◁

One possible approach would be to keep all instantiated non-temporal terms unchanged, but replace them by variables if a new query on the same predicate is generated. We defer the concrete choice of strategy to future work, as it is immaterial for this work. In particular, the previous comment on Propositions 10 and 12 applies in either case.

Theorem 6 (Soundness and completeness of pre-processing). *Let* $Q^* = \langle \Pi, P\rangle$ *be one of the queries obtained from the pre-processing of* $Q$*,* $F$ *be a set of literals with timestamp higher than* $\tau$ *such that* $F^+$ *only contains EDB atoms. Assume that* $\Pi \cup D_\tau \cup F$ *is consistent, and let* $\theta$ *be a substitution that instantiates all variables in* $P$*. Then* $\Pi \cup D_\tau \cup F \models P\theta$ *iff there exist substitutions* $\theta'$ *and* $\sigma$ *and a set* $H$ *such that* $\langle \theta', H\rangle \in \mathcal{P}_{Q^*}$*,* $\theta = \theta'\sigma$ *and* $H\sigma \subseteq F$*.*

Proof. From soundness and completeness of SLD-resolution, it immediately follows that $\Pi \cup D_\tau \cup F \models P\theta$ iff $\langle \theta', H\rangle \in \mathcal{P}_Q$ for some $\theta'$ and $H$ such that $\theta = \theta'\sigma$ and $H\sigma \subseteq F$ for some substitution $\sigma$. (Note that the pre-processing step essentially treats negated literals as EDB atoms.) □

### 6.4 The iterative step

In the online step of the algorithm, we now need to compute schematic answers not only to the original query, but also to each query generated by the pre-processing step. In this section, we use $Q^*$ to range over all these queries. This step is now significantly more complex than for the language without negation: besides updating the (positive) hypotheses in hypothetical answers with information from the data stream, we need to apply a fixpoint construction in order to update the negative hypotheses – these are updated using the computed schematic answers themselves, and each update can trigger new possible updates.

In order to simplify the presentation, we first formulate the iterative step working only with ground terms. After discussing its soundness and completeness, we show how it can be reformulated in terms of schematic hypothetical answers using variables, as in the previous case. Without loss of generality, we also assume that all constants are abstracted to variables in the queries generated by the pre-processing step, so that for each literal $\ell$ there exists at most one query on the predicate symbol in $\ell$. (This assumption removes the need for some quantifications, but does not change anything in an essential manner.)

We work with *generalized hypothetical answers*, where hypotheses can be EDB literals or negated IDB atoms, and we recursively define sets $\mathcal{S}^{\downarrow}_\tau(Q^*)$ of these entities. The definition uses an auxiliary family $\mathcal{A}_\tau(Q^*)$; intuitively, $\mathcal{A}_\tau(Q^*)$ is computed from $\mathcal{S}^{\downarrow}_{\tau-1}(Q^*)$ by updating positive hypotheses according to the data stream (as in the case without negation), and $\mathcal{S}^{\downarrow}_\tau(Q^*)$ is computed from $\mathcal{A}_\tau(Q^*)$ by updating the negative hypotheses.

Definition 21. *The families of sets $\mathcal{S}^{\downarrow}_{\tau}(Q^*)$ and $\mathcal{A}_{\tau}(Q^*)$ are defined recursively as follows, where $Q^* = \langle P, \Pi \rangle$ is one of the queries under consideration.*

*(1)* $\mathcal{S}^{\downarrow}_{-1}(Q^*) = \emptyset$.

*(2) For each $\langle \theta, E, H \rangle \in \mathcal{S}^{\downarrow}_{\tau-1}Q^*$, let $M$ be the (possibly empty) set of elements of $H^+$ with timestamp $\tau$. If $M \subseteq D|_{\tau}$, $\langle \theta, E \cup M, H \setminus M \rangle \in \mathcal{A}_{\tau}(Q^*)$.*

*(3) For each $\langle \theta, H \rangle \in \mathcal{P}_{Q^*}$, let $M$ be the set of elements of $H$ with minimal timestamp. For each substitution $\sigma$ such that $H\sigma$ is ground:*

*(a) If $M^+$ is non-empty and $M^+\sigma \subseteq D|_{\tau}$, then $\langle \theta\sigma, M^+\sigma, (H \setminus M^+)\sigma \rangle \in \mathcal{A}_{\tau}(Q^*)$.*

*(b) If $P\sigma$ has timestamp $\tau$ and all elements of $M\sigma$ are negative literals, then $\langle \theta\sigma, \emptyset, H\sigma \rangle \in \mathcal{A}_{\tau}(Q^*)$.*

*(c) If $P\sigma$ has timestamp $\tau$ and every element of $H\sigma$ has timestamp strictly larger than $\tau$, then $\langle \theta\sigma, \emptyset, H\sigma \rangle \in \mathcal{A}_{\tau}(Q^*)$.*

*(4) $\mathcal{S}^{\downarrow}_{\tau}(Q^*)$ is obtained by applying the following operator $\mathcal{R}$ to $\mathcal{A}_{\tau}(Q^*)$ until a fixpoint is reached: given a family $X(Q^*)$ of generalized hypothetical answers, non-deterministically select a query $Q_\ell$ over a literal $\ell$.*

*(a) If there is a tuple $\langle \theta, E_\ell, \emptyset \rangle \in X(Q_\ell)$, then for all $Q^*$ remove every $\langle \sigma, E, H \rangle$ from $X(Q^*)$ whenever $\ell\theta \in H^-$.*

*(b) For all substitutions $\sigma$ such that the timestamp of $\ell\sigma$ is at most $\tau$ and there is no tuple $\langle \sigma, E_\ell, H_\ell \rangle \in X(Q_\ell)$, if $X(Q^*)$ contains an element $\langle \theta, E, H \rangle$ such that $\ell\sigma \in H^-$, then remove $\neg\ell\sigma$ from $H$ and add it to $E$.*

*We assume that $\mathcal{R}$ selects a query $Q_\ell$ such that $X$ changes, if possible.*

The operator $\mathcal{R}$ randomly selects a query $Q_\ell$ and updates the generalized hypothetical answers that use negative ground instances of literal $\ell$ by (i) discarding the ones whose hypotheses contradict generated answers to $Q_\ell$ and (ii) moving hypotheses to evidence when they match no generalized hypothetical answer to $Q_\ell$. In order to guarantee that all these steps are finite, the second step in the computation of $\mathcal{R}(X)$ only looks at hypotheses whose timestamp is at most the current value of $\tau$ – steps 3(b) and 3(c) ensure that $\mathcal{R}$ does not incorrectly remove some negative hypotheses.

Lemma 15. *Iterating $\mathcal{R}$ always reaches a fixpoint in finite time.*

Proof. If $\mathcal{R}(X)$ is different from $X$, then either the total number of elements in all the sets $X(Q^*)$ decreases or the total number of elements in all the sets $H$ in tuples $\langle \theta, E, H \rangle$ in all $X(Q^*)$ decreases. But set $X$ is finite: every element of $\mathcal{A}_{\tau}(Q^*)$ either comes from unification between $\mathcal{P}_{Q^*}$ and the current $\tau$-history (and since both $\mathcal{P}_{Q^*}$ and $\tau$-histories are finite, only a finite number of elements can be generated in this way), or from directly instantiating elements of $\mathcal{P}_{Q^*}$ with a timestamp at most $\tau$ (which can only be done in finitely many ways). Therefore this process must terminate. □

Lemma 16. *For each query $Q^*$, the set $\mathcal{S}^{\downarrow}_{\tau}(Q^*)$ is uniquely defined.*

Proof. We start by showing that: if we mark the generalized hypothetical answers that can be removed from a set $X$ and the set of (negative) hypotheses that can be turned into evidence, then, after applying $\mathcal{R}$ once, all actions that were not made can still be applied afterwards. Together with the fact that the actions enabled by new tuples that arrive are the same, this implies that all actions must have been performed when reaching a fixpoint.

Let $X$ and $Y$ be families of generalized hypothetical answers to the queries $Q^*$ such that $Y$ can be obtained from $X$ by application of $R$.

- If $\langle \sigma, E, H \rangle$ can be removed from $X(Q^*)$ by application of $\mathcal{R}$ and there is a corresponding tuple $\langle \sigma, E', H' \rangle$ in $Y$ (either the same or obtained by moving something from $H$ to $E$ in the second case defining $\mathcal{R}$), then $\langle \sigma, E', H' \rangle$ can be removed from $Y(Q^*)$ by application of $\mathcal{R}$.

Indeed, if $\langle \theta, E_\ell, \emptyset \rangle \in X(Q_\ell)$, then this answer must still be in $Y(Q_\ell)$, since $\mathcal{R}$ does not remove answers (with empty set of hypotheses). Furthermore, if $\ell\theta \in H^-$ then $\ell\theta \in H'^-$, since $\langle \theta, E_\ell, \emptyset \rangle$ prevents $\ell\theta$ from being removed from $H$ by application of $\mathcal{R}$.

- If $\langle \sigma, E, H \rangle \in X(Q^*)$ can be transformed into $\langle \sigma, E \cup \{\ell\theta\}, H \setminus \{\ell\theta\} \rangle$ by application of $\mathcal{R}$ and there is a corresponding tuple $\langle \sigma, E', H' \rangle$ in $Y$ with $\ell\theta \in H'$, then $\langle \sigma, E', H' \rangle$ can be changed to $\langle \sigma, E' \cup \{\ell\theta\}, H' \setminus \{\ell\theta\} \rangle$ by application of $\mathcal{R}$.
  Indeed, if the timestamp of $\ell\theta$ is at most $\tau$ and there is no tuple $\langle \theta, E, H \rangle \in X(Q_\ell)$, then this must still be the case in $Y(Q_\ell)$, since $\mathcal{R}$ does not add new tuples to $X$.

Now consider two sequences of applications of $\mathcal{R}$ starting from $X$ and ending at a fixpoint. By inductively applying the previous argument, any tuples that can be removed or changed in $X$ must have been removed or changed in the same way when reaching the fixpoint. Furthermore, new tuples that can be removed or changed due to tuples that were removed or changed will be the same in both sequences of applications, since these tuples are directly determined by the action(s) performed. Therefore both sequences must end at the same fixpoint. □

We illustrate this construction by means of a small example, before moving to our richer running example.

Example 19. *Consider the query $Q = \langle \mathsf{R}(T), \Pi' \rangle$ where $\Pi'$ is the following program.*

$$\mathsf{S}(T), \mathsf{S}(T+1) \longrightarrow \mathsf{P}(T) \qquad\qquad \neg\mathsf{P}(T) \longrightarrow \mathsf{R}(T)$$

*The pre-processing step for this query generates $\mathcal{P}_Q = \{\langle \emptyset, \{\neg\mathsf{P}(T)\}\rangle\}$. Since $\mathsf{P}(T)$ occurs negated in the premise of the only answer in this set, we generate a new query $Q' = \langle \mathsf{P}(T), \Pi \rangle$, for which the pre-processing step yields $\mathcal{P}_{Q'} = \{\langle \emptyset, \{\mathsf{S}(T), \mathsf{S}(T+1)\}\rangle\}$.*

*Assume that $D_0 = \emptyset$. Since the only element of $P_Q$ contains a negated atom in its premises, we are in case (3c) and $\mathcal{A}_0(Q) = \langle [T := 0], \emptyset, \{\neg\mathsf{P}(0)\}\rangle$. Furthermore, $\mathcal{A}_0(Q') = \emptyset$, since $\mathsf{S}(0) \notin D_0$. Since there is no element in $\mathcal{A}_0(Q')$, we move $\neg\mathsf{P}(0)$ to the set of evidence in the only element of $\mathcal{A}_0(Q)$; this yields a fixpoint of $\mathcal{R}$, so $\mathcal{S}_0(Q) = \langle [T := 0], \{\neg\mathsf{P}(0)\}, \emptyset \rangle$ and $\mathcal{S}_0(Q') = \emptyset$.*

*Now suppose that $D_1 = \{\mathsf{S}(1)\}$. Reasoning as before, we obtain $\mathcal{A}_1(Q) = \{\langle [T := 1], \emptyset, \{\neg\mathsf{P}(1)\}\rangle\}$. However, since $\mathsf{S}(1) \in D_1$, we also obtain $\mathcal{A}_1(Q') = \{\langle [T := 1], \{\mathsf{S}(1)\}, \{\mathsf{S}(2)\}\rangle\}$. Now there is an answer in $\mathcal{A}_1(Q')$ with substitution $[T := 1]$ and non-empty set of premises, so we cannot conclude anything about $\mathsf{P}(1)$ yet, and thus $\mathcal{A}_1$ is already a fixpoint of $\mathcal{R}$; so $\mathcal{S}_1 = \mathcal{A}_1$.*

*If $D_2 = \{\mathsf{S}(2)\}$, then $\langle [T := 1], \{\mathsf{S}(1), \mathsf{S}(2)\}, \emptyset \rangle \in \mathcal{A}_2(Q')$. As a consequence, $\neg\mathsf{P}(1)$ does not hold, and $\langle [T := 1], \emptyset, \{\neg\mathsf{P}(1)\}\rangle \in \mathcal{A}_2(Q)$ is removed by $\mathcal{R}$.*

*On the other hand, if $D_2 = \emptyset$, it is the answer $\langle [T := 1], \{\mathsf{S}(1)\}, \{\mathsf{S}(2)\}\rangle \in \mathcal{S}_1(Q')$ that does not propagate to $\mathcal{A}_2(Q')$, from which applying $\mathcal{R}$ will allow us to conclude that $[T := 1]$ is output as an answer to $Q$.* ◁

Soundness and completeness hold for $T$-stratified programs. Proving soundness requires a partial form of completeness, which is why the next lemma is stated as an if-and-only-if.

Theorem 7 (Soundness). *Let $Q^* = \langle \Pi, P \rangle$, where $\Pi$ is $T$-stratified, be one of the queries obtained from the pre-processing of $Q$, $F$ be a set of literals with timestamp higher than $\tau$ such that $F^+$ only contains EDB atoms. Assume that $\Pi \cup D_\tau \cup F$ is consistent, and let $\theta$ be a substitution that instantiates all variables in $P$.*

*Then:*

*(i) If $\langle \theta, E, H \rangle \in \mathcal{S}_\tau^{\downarrow}(Q^*)$ for some $E$ and $H$ such that $E^+ \subseteq D_\tau$ and $H \subseteq F$, then $\Pi \cup D_\tau \cup F \models P\theta$.*
*(ii) If the timestamp of $P\theta$ is at most $\tau$, then the converse implication also holds.*

Proof. The proof is by induction on the construction of $\mathcal{S}^{\downarrow}_{\tau}(Q^*)$, considering all queries $Q^*$ simultaneously. For clarity, we prove (i) and (ii) separately arguing by induction on the timestamp, but their proofs use each other's induction hypothesis. We point out that induction hypotheses are always applied to sets that have been constructed earlier, i.e. we first conclude (i) and (ii) for $\mathcal{S}^{\downarrow}_{-1}$, then for $\mathcal{A}_0$, then for $\mathcal{R}(\mathcal{A}_0)$, then for $\mathcal{R}(\mathcal{R}(\mathcal{A}_0))$, etc, until we reach $\mathcal{S}^{\downarrow}_{0}$.

We first prove property (i). For $\tau = -1$, this is trivial. Assume that the result holds for $\mathcal{S}_{\tau-1}$. We go through the steps of the construction of $\mathcal{S}_{\tau}(Q^*)$.

(Step 2.) Suppose that $\langle \theta, E, H\rangle \in \mathcal{S}^{\downarrow}_{\tau-1}(Q^*)$, that $M \subseteq D|_\tau$ for the set $M$ of elements with timestamp $\tau$ of $H$, and that $E^+ \cup M \subseteq D_\tau$ and $H \setminus M \subseteq F$. By construction $E^+ \subseteq D_{\tau-1}$, and $H \subseteq F \cup M \subseteq F \cup D|_\tau$. If $\Pi \cup D_\tau \cup F$ is consistent, then the induction hypothesis yields the thesis, since $\Pi \cup D_\tau \cup F = \Pi \cup D_{\tau-1} \cup (F \cup D|_\tau)$.

(Step 3.) Suppose that $\langle \theta, H\rangle \in \mathcal{P}_{Q^*}$, let $M$ be the set of elements in $H$ with minimal timestamp, and suppose that $\sigma$ makes all elements of $H\sigma$ ground. The cases (b) and (c) follow immediately from Theorem 6.
For case (a), suppose that $M^+\sigma \subseteq D|_\tau$ and that $(H \setminus M^+)\sigma \subseteq F$. In particular, all elements of $M$ have timestamp $\tau$. Then $H\sigma \subseteq F \cup M^+\sigma \subseteq F \cup D|_\tau$, so by Theorem 6, $\Pi \cup D_{\tau-1} \cup (F \cup D|_\tau) \models P\theta\sigma$. The thesis follows by observing that $\Pi \cup D_{\tau-1} \cup (F \cup D|_\tau) = \Pi \cup D_\tau \cup F$.

(Step 4.) The proof follows by induction on the number of times that $\mathcal{R}$ is applied. Again, step (a) is trivial, since it generates no new generalized hypothetical answers.
For step (b), we need to look at the generalized hypothetical answers that are changed by $\mathcal{R}$. Assume that $\langle \theta, E, H\rangle$ is such that $\ell\sigma \in H^-$ for some $\ell$ and $\sigma$ with the timestamp of $\ell\sigma$ at most $\tau$, and that there is no tuple $\langle \sigma, E_\ell, H_\ell\rangle \in X(Q_\ell)$. Suppose that $(E \cup \{\neg\ell\sigma\})^+ \subseteq D_\tau$ and $H \setminus \neg\ell\sigma \subseteq F$. Then $E^+ \subseteq D_\tau$ and $H \subseteq F \cup \{\neg\ell\sigma\}$.
But $\Pi \cup D_\tau \cup (F \cup \{\neg\ell\sigma\})$ is consistent iff $\Pi \cup D_\tau \cup F \not\models \ell\sigma$, which by induction hypothesis (on $Q_\ell$) is the case since there are no tuples $\langle \sigma, E_\ell, H_\ell\rangle \in X(Q_\ell)$. Therefore the induction hypothesis (on $Q^*$) allows us to conclude that $\Pi \cup D_\tau \cup (F \cup \{\neg\ell\sigma\}) \models P\theta$. But the induction hypothesis (on $Q_\ell$) also yields that $\Pi \cup D_\tau \cup F \models \neg\ell\sigma$: since $\Pi$ is stratified, if this were not the case then $\Pi \cup D_\tau \cup F \models \ell\sigma$, whence there would be some tuple $\langle \sigma, E_\ell, H_\ell\rangle \in X(Q_\ell)$.

We now move to property (ii). From Theorem 6, there exists $\langle \theta', H_0\rangle \in \mathcal{P}_{Q^*}$ such that $\theta = \theta'\sigma$ for some substitution $\sigma$ and $H_0\sigma \subseteq F$. We prove that (ii) holds whenever the minimal timestamp of $H_0^+\sigma \cup \{P\theta\}$ is at most $\tau$, which implies the thesis. For $\tau = -1$ this vacuously holds, since timestamps must be positive.

Assume that $\Pi \cup D_\tau \cup F \models P\theta$. If the minimal timestamp of $H_0^+\sigma \cup \{P\theta\}$ is strictly smaller than $\tau$, then by induction hypothesis applied to $F \cup D|_\tau$, there exists $\langle \theta, E, H\rangle \in \mathcal{S}^{\downarrow}_{\tau-1}(Q^*)$ such that $E^+ \subseteq D_{\tau-1}$ and $H \subseteq F \cup D|_\tau$. Step (2) of the algorithm then adds $\langle \theta, E \cup M, H \subseteq M\rangle$ to $\mathcal{A}_\tau(Q^*)$, where $M \subseteq D|_\tau$, and by construction $(E \cup M)^+ \subseteq D_\tau$ and $H \subseteq F$. If the minimal timestamp of $H_0^+\sigma \cup \{P\theta\}$ is exactly $\tau$, then a tuple satisfying the thesis is added to $\mathcal{A}_\tau(Q^*)$ in step (3).

We now show that the fixpoint construction in step (4) does not break this property.

- Suppose step 4(a) applies to a set $X$. If there is a tuple $\langle \theta, E_\ell, \emptyset\rangle \in X(Q_\ell)$, then by induction hypothesis $\Pi \cup D_\tau \cup F' \models \ell\theta$ for any set $F'$ in the conditions of the theorem, and therefore $\Pi \cup D_\tau \cup F$ is inconsistent whenever $\neg\ell\theta \in F$. As such, any tuple whose set $H$ contains $\neg\ell\theta$ can be removed from $X$.
- Suppose step 4(b) applies to a set $X$. Since this step adds elements to $E^-$ and removes elements from $H$, the thesis holds even if this step is applied to this tuple. □

This result guarantees that any answer to query $Q$ will eventually be represented by an element of $\mathcal{S}^{\downarrow}_{\tau}(Q)$, but it does not ensure that that element necessarily becomes an answer, since the set of premises may never become empty. The next result shows that this is indeed the case.

Proposition 17. *In the same conditions as the previous theorem, if $\langle \theta, E, H\rangle \in \mathcal{S}^{\downarrow}_{\tau}(Q^*)$, then there exists $\tau' \geq \tau$ such that either (a) $\langle \theta, E \cup H, \emptyset\rangle \in \mathcal{S}^{\downarrow}_{\tau'}(Q^*)$ or (b) $\mathcal{S}^{\downarrow}_{\tau'}(Q^*)$ does not contain any element of the form $\langle \theta, E', H'\rangle$ such that $E' \cup H' = E \cup H$.*

Proof. By induction on the stratification of $\Pi$, i.e., on the stratum of $P_t$, where $t$ is the (instantiated) temporal argument in $P\theta$.

If $P_t$ is in the lowest stratum, then it does not depend on any negated predicates. The thesis then follows by observing that the algorithm behaves for this predicate as in the case of the fragment without negation, and invoking Theorems 3 and 4.

Otherwise, by induction hypothesis, for every negative literal in $H$ there is a timestamp where the corresponding query is decided (any schematic answer for the relevant predicate has no premises). Take $\tau_0$ to be the highest of those timestamps. In step (4) of the algorithm, in any generalized hypothetical answers where $\theta$ instantiates the timestamp of $P$ to $t$, all negated literals in the set of premises must either lead to the answer being discarded 4(a), or be moved to the set of evidence 4(b). Therefore, at the end of this iteration, for any generalized hypothetical answer $\langle \theta, E', H'\rangle \in \mathcal{S}^{\downarrow}_{\tau_0+1}(Q^*)$ the set $H'$ only has positive literals. If $H'$ is always empty, the thesis holds. Otherwise, let $\tau'$ be the highest timestamp in all such sets $H'$; the construction in step (3) guarantees that in $\mathcal{S}^{\downarrow}_{\tau'}(Q^*)$ all $H'$ in answers that have not been discarded must be empty. □

Corollary 18 (Completeness). *If $\theta$ is an answer to $Q$, then there exists $\tau$ such that $\mathcal{S}_{\tau}(Q)$ contains a tuple $\langle \theta', E, \emptyset\rangle$ where $\theta' = \theta\sigma$ for some substitution $\sigma$.*

*Remark.* The correspondence with the declarative notion of (supported) hypothetical answer is less direct, since generalized hypothetical answers are allowed to have negated literals. Variants of Theorem 7 and Lemma 17 relating directly to hypothetical answers can be obtained by recursively processing negative hypotheses and evidence:

- if $\neg\ell$ appears in a set of evidence, replace it with the evidence for $\ell$ that was used to place it there in step 4(b);
- if $\neg\ell\theta$ appears in a set of hypotheses, then replace the generalized hypothetical answer with all possible generalized hypothetical answers where $\neg\ell$ is replaced by $\neg\ell'$ with $\langle \theta, E, H\rangle \in \mathcal{S}_{\tau}(Q^{\ell})$ and $\ell' \in H$.

The last step generates a combinatorial explosion of the number of hypothetical answers. We also believe that generalized hypothetical answers are more readable in practice, as they encapsulate the strata of the program (and the information about negative hypotheses/evidence can be inspected separately). For these reasons, we do not pursue this correspondence further.

We now present an alternative to Definition 21 avoiding grounding and discuss its complexity.

Definition 22. *The sets $\mathcal{S}_{\tau}(Q^*)$ and $\mathcal{B}_{\tau}(Q^*)$, where $Q^* = \langle P, \Pi\rangle$ is one of the queries under consideration, are recursively defined as follows.*

*(1) $\mathcal{S}_{-1}(Q^*) = \emptyset$.*

*(2) For each $\langle \theta, E, H\rangle \in \mathcal{S}_{\tau-1}(Q^*)$, let $M$ be the (possibly empty) set of elements of $H^+$ with timestamp $\tau$. For all substitutions $\sigma$ such that $M\sigma \subseteq D|_{\tau}$, $\langle \theta, E \cup M\sigma, (H \setminus M)\sigma\rangle \in \mathcal{B}_{\tau}(Q^*)$.*

*(3) For each $\langle \theta, H\rangle \in \mathcal{P}_{Q^*}$, let $M$ be the set of elements of $H$ with minimal timestamp.*

*(a) If $M^+$ is non-empty, then for every substitution $\sigma$ such that $M^+\sigma \subseteq D|_{\tau}$, then $\langle \theta\sigma, M^+\sigma, (H \setminus M^+)\sigma\rangle \in \mathcal{B}_{\tau}(Q^*)$.*

*(b) Let $\sigma = [T := \tau]$ with $T$ the temporal variable in $P$. If all elements of $M\sigma$ are negative literals, then $\langle \theta\sigma, \emptyset, H\sigma \rangle \in \mathcal{B}_\tau(Q^*)$.*

*(c) Let $\sigma = [T := \tau]$ with $T$ the temporal variable in $P$. If every element of $M\sigma$ has timestamp strictly larger than the timestamp of $P\sigma$, then $\langle \theta\sigma, \emptyset, H\sigma \rangle \in \mathcal{B}_\tau(Q^*)$.*

*(4) $\mathcal{S}_\tau(Q^*)$ is obtained from $\mathcal{B}_\tau(Q^*)$ as follows. Fix a topological ordering of the stratification of $\Pi^\downarrow$. There is only a finite number of $P_t$ such that $Q_P$ has at least one schematic answer with $T\theta = t$ (where $T$ is the temporal parameter in $P$). For each of these $P_t$ in order, set $\ell = P(t_1, \ldots, t_n)$ and let $S(P_t)$ be the set of schematic answers $\langle \theta, E, H \rangle$ to $Q_P$ such that $T\theta = t$.*

*(a) if $S(P_t)$ contains a tuple $\langle \theta, E, \emptyset \rangle$, then: in each $Q^*$, replace every $\langle \sigma, E, H \rangle$ such that $\ell\theta$ is unifiable with an element $h \in H^-$ with all possible $\langle \sigma\theta', E\theta', H\theta' \rangle$ such that $\theta'$ is a minimal substitution with the property that $\ell\theta$ is not unifiable with $h\theta'$;*

*(b) in each $\mathcal{B}_\tau(Q^*)$, for each $\langle \theta, E, H \rangle$ such that $H^-$ contains an element $h$ with predicate symbol $P$ and timestamp $t$ and there is no tuple $\langle \sigma', E_\ell, H_\ell \rangle \in S(P_t)$ such that $h$ and $\ell\sigma'$ are unifiable, then remove $\neg h$ from $H$ and add it to $E$.*

Lemma 19. *The elements of $\mathcal{S}^\downarrow_\tau(Q^*)$ are exactly the ground instances of the elements of $\mathcal{S}_\tau(Q^*)$.*

Proof (Sketch.) Straightforward by induction on the construction of the sets $\mathcal{S}^\downarrow_\tau$ and $\mathcal{S}_\tau$, observing that the steps in Definitions 21 and 22 always yield sets in the relation stated in the lemma.

For step 4, since the order of iterations of $\mathcal{R}$ to compute $\mathcal{S}^\downarrow_\tau$ from $\mathcal{A}_\tau$ is immaterial (Lemma 16), we can assume without loss of generality that $\mathcal{R}$ always chooses a literal in the lowest possible stratum. Thus one application of $\mathcal{R}$ (Definition 21) corresponds to performing step 4 in Definition 22 for one literal $\ell$. Finally, using a topological ordering guarantees that a fixpoint is reached, since processing one of the $P_t$ cannot change elements of $S(P'_t)$ for each $P'_t$ previously processed. □

Theorem 8 (Complexity). *Let $k$ be the highest arity of any predicate that occurs negated in $\Pi$. If $\Pi$ is $T$-stratified, then $\mathcal{S}_\tau$ can be computed in time exponential in $k$ and polynomial on the size of $\mathcal{P}_Q$, $\mathcal{S}_{\tau-1}$, $D|_\tau$ and the total number of queries.*

Proof. Steps (1), (2) and (3a) are as in Theorem 5, but they are now iterating over all sets of queries. Therefore the previous time bounds apply, multiplied by the total number of queries. Steps (3b) and (3c) can be done in time linear on the size of $\mathcal{P}_Q$.

Each iteration of step (4) requires computing the set of negative literals in a schematic answer (which can be done in time linear on the size of the answer) and checking whether a corresponding tuple exists in the current $\mathcal{B}_\tau$. This can be done in time linear in the size of $\mathcal{B}_\tau$, which is polynomial in the size of $\mathcal{S}_{\tau-1}$ (since it was computed from $\mathcal{S}_{\tau-1}$ by a polynomial algorithm). The number of substitutions $\sigma$ that need to be considered is bound by the number of constants to the power $k$, which is constant; thus any remaining answers can be added in worst-case polynomial time. Finally, the total number of iterations of step (4) is at most the total number of queries multiplied by $\tau$. □

The only parameter on which there is an exponential dependency in this complexity bound is $k$. In practice, this value is small, since predicates in real-life programs typically do not involve more than a few variables. This minimizes the impact of this exponential complexity.

We can also improve on the average-time complexity by delaying the instantiation in step 4 through a number of other techniques. One possibility is delaying the application of step 4 until there are no positive atoms in $H$, so that all negative literals are as instantiated as possible. In particular, if all negations in $\Pi$ are safe, the only potential uninstantiated variables are those in the original query $Q$. Another possibility is storing $\sigma$ abstractly, e.g. as a set of constraints on the substitution $\theta$ (such as $[X \neq \mathrm{a}]$). In the worst case, though, any of these approaches still has the

same complexity as the simpler version presented earlier – since we can always find an example where the number of answers to the original query is exponential on $k$.

We now illustrate this algorithm with our running example, where all cases are covered.

Example 20. *Assume that there are two patients being monitored,* john *and* gus. *Recall that preprocessing yielded the sets*

$$\begin{aligned}
\mathcal{P}_{Q_R} &= \{\langle \emptyset, \{\mathsf{BCA}(X, T+2), \underline{\neg\mathsf{GVS}(X, T+1)}\}\rangle,\\
&\qquad \langle \emptyset, \{\neg\mathsf{GVS}(X, T+1), \underline{\neg\mathsf{GVS}(X, T), \neg\mathsf{ST}(X, \mathsf{us}, T)}\}\rangle\\
\mathcal{P}_{Q_G} &= \langle \emptyset, \{\underline{\mathsf{GCM}(X, T), \mathsf{GBOL}(X, T)}\}\rangle\\
\mathcal{P}_{Q_S} &= \{\langle \emptyset, \{\underline{\mathsf{BCA}(X, T+1)}\}\rangle,\\
&\qquad \langle \emptyset, \{\underline{\neg\mathsf{GVS}(X, T-1), \neg\mathsf{ST}(X, \mathsf{us}, T-1)}\}\rangle
\end{aligned}$$

*where the literals with minimal timestamp in each set of premises are underlined, and that, by definition,* $\mathcal{S}_{-1}(Q_R) = \mathcal{S}_{-1}(Q_G) = \mathcal{S}_{-1}(Q_S) = \emptyset$.

*We start at time point* $\tau = 0$*, and assume that* $D|_0 = \{\mathsf{GCM}(\mathsf{gus}, 0)\}$. *Since* $\mathcal{S}_{\tau-1}(Q^*) = \emptyset$ *for all queries* $Q^*$*, step 2 is trivial.*

*The table below shows the schematic hypothetical answers added in each substep of step 3. Substeps 3(b) and 3(c) uses* $\sigma = [T := 0]$ *in all queries.*

| *Step* | $\mathcal{B}_0(Q_R)$ | $\mathcal{B}_0(Q_G)$ | $\mathcal{B}_0(Q_S)$ |
|---|---|---|---|
| *3(a)* | *nothing* | *nothing* | *nothing* |
| *3(b)* | $\langle \sigma, \emptyset, \{\mathsf{BCA}(X, 2), \neg\mathsf{GVS}(X, 1)\}\rangle$<br>$\langle \sigma, \emptyset, \{\neg\mathsf{GVS}(X, 1), \neg\mathsf{GVS}(X, 0), \neg\mathsf{ST}(X, \mathsf{us}, 0)\}\rangle$ | *nothing* | *nothing* |
| *3(c)* | $\langle \sigma, \emptyset, \{\mathsf{BCA}(X, 2), \neg\mathsf{GVS}(X, 1)\}\rangle$ | *nothing* | $\langle \sigma, \emptyset, \{\mathsf{BCA}(X, 1)\}\rangle$ |

*We obtain:*

$$\begin{aligned}
\mathcal{B}_0(Q_R) &= \{\langle [T := 0], \emptyset, \{\mathsf{BCA}(X, 2), \neg\mathsf{GVS}(X, 1)\}\rangle,\\
&\qquad \langle [T := 0], \emptyset, \{\neg\mathsf{GVS}(X, 1), \neg\mathsf{GVS}(X, 0), \neg\mathsf{ST}(X, \mathsf{us}, 0)\}\rangle\}\\
\mathcal{B}_0(Q_G) &= \emptyset\\
\mathcal{B}_0(Q_S) &= \{\langle [T := 0], \emptyset, \{\mathsf{BCA}(X, 1)\}\rangle\}
\end{aligned}$$

*We move on to step 4. In the stratification (up to timestamp* 0*),* $\mathsf{GVS}_0$ *and* $\mathsf{ST}_0$ *at the lowest stratum, and* $\mathsf{Risk}_0$ *depends on both of them; we include only the relevant information on each* $S(P_t)$ *– the substitution and whether the set of hypotheses is empty. Since* $\mathsf{Risk}_0$ *is at the highest stratum, no schematic hypothetical answers can depend on it, and we do not include it in the table.*

| $P_t$ | $S(P_t)$ | *set* | *tuples of interest* |
|---|---|---|---|
| $\mathsf{GVS}_0$ | $\emptyset$ | $\mathcal{B}_0(Q_R)$ | $\langle [T := 0], \emptyset, \{\neg\mathsf{GVS}(X, 1), \neg\mathsf{GVS}(X, 0), \neg\mathsf{ST}(X, \mathsf{us}, 0)\}\rangle$<br>*becomes*<br>$\langle [T := 0], \{\neg\mathsf{GVS}(X, 0)\}, \{\neg\mathsf{GVS}(X, 1), \neg\mathsf{ST}(X, \mathsf{us}, 0)\}\rangle$ |
| $\mathsf{ST}_0$ | $[T := 0]$<br>$H \neq \emptyset$ | $\mathcal{B}_0(Q_R)$ | $\langle [T := 0], \{\neg\mathsf{GVS}(X, 0)\}, \{\neg\mathsf{GVS}(X, 1), \neg\mathsf{ST}(X, \mathsf{us}, 0)\}\rangle$<br>*unchanged* |

*We finally obtain:*

$$\begin{aligned}\mathcal{S}_0(Q_R) = \{&\langle [T := 0], \emptyset, \{\mathsf{BCA}(X, 2), \neg\mathsf{GVS}(X, 1)\}\rangle,\\ &\langle [T := 0], \{\neg\mathsf{GVS}(X, 0)\}, \{\neg\mathsf{GVS}(X, 1), \neg\mathsf{ST}(X, \mathsf{us}, 0)\}\rangle\}\\ \mathcal{S}_0(Q_G) = \emptyset\\ \mathcal{S}_0(Q_S) = \{&\langle [T := 0], \emptyset, \{\mathsf{BCA}(X, 1)\}\rangle\}\end{aligned}$$

*We move on to $\tau = 1$, and assume that $D|_1 = \{\mathsf{BCA}(\mathsf{john}, 1), \mathsf{GCM}(\mathsf{gus}, 1)\}$. We summarize the application of step 2 in the next table.*

| *set* | *tuple* |
|---|---|
| $\mathcal{B}_1(Q_R)$ | $\langle [T := 0], \emptyset, \{\mathsf{BCA}(X, 2), \neg\mathsf{GVS}(X, 1)\}\rangle$ *(unchanged from $\mathcal{S}_0(Q_R)$)* |
| $\mathcal{B}_1(Q_R)$ | $\langle [T := 0], \{\neg\mathsf{GVS}(X, 0)\}, \{\neg\mathsf{GVS}(X, 1), \neg\mathsf{ST}(X, \mathsf{us}, 0)\}\rangle$ *(unchanged from $\mathcal{S}_0(Q_R)$)* |
| $\mathcal{B}_1(Q_S)$ | $\langle [T := 0, X := \mathsf{john}], \{\mathsf{BCA}(\mathsf{john}, 1)\}, \emptyset\rangle$ *(from unifying $\mathsf{BCA}(X, 1)$ with $D\|_1$)* |

*Furthermore, in step 3 we obtain the following additional schematic hypothetical answers, where substeps 3(b) and 3(c) use $\sigma = [T := 1]$ in all queries.*

| *Step* | $\mathcal{B}_1(Q_R)$ | $\mathcal{B}_1(Q_G)$ | $\mathcal{B}_1(Q_S)$ |
|---|---|---|---|
| *3(a)* | *nothing* | *nothing* | $\langle [T := 0, X := \mathsf{john}], \{\mathsf{BCA}(\mathsf{john}, 1)\}, \emptyset\rangle$ |
| *3(b)* | $\langle \sigma, \emptyset, \{\mathsf{BCA}(X, 3), \neg\mathsf{GVS}(X, 2)\}\rangle$<br>$\langle \sigma, \emptyset, \{\neg\mathsf{GVS}(X, 2), \neg\mathsf{GVS}(X, 1), \neg\mathsf{ST}(X, \mathsf{us}, 1)\}\rangle$ | *nothing* | $\langle \sigma, \emptyset, \{\neg\mathsf{GVS}(X, 0), \neg\mathsf{ST}(X, \mathsf{us}, 0)\}\rangle$ |
| *3(c)* | $\langle \sigma, \emptyset, \{\mathsf{BCA}(X, 3), \neg\mathsf{GVS}(X, 2)\}\rangle$ | *nothing* | $\langle \sigma, \emptyset, \{\mathsf{BCA}(X, 2)\}\rangle$ |

*Combining all these results, we obtain:*

$$\begin{aligned}\mathcal{B}_1(Q_R) = \{&\langle [T := 0], \emptyset, \{\mathsf{BCA}(X, 2), \neg\mathsf{GVS}(X, 1)\}\rangle,\\ &\langle [T := 0], \{\neg\mathsf{GVS}(X, 0)\}, \{\neg\mathsf{GVS}(X, 1), \neg\mathsf{ST}(X, \mathsf{us}, 0)\}\rangle,\\ &\langle [T := 1], \emptyset, \{\mathsf{BCA}(X, 3), \neg\mathsf{GVS}(X, 2)\}\rangle,\\ &\langle [T := 1], \emptyset, \{\neg\mathsf{GVS}(X, 2), \neg\mathsf{GVS}(X, 1), \neg\mathsf{ST}(X, \mathsf{us}, 1)\}\rangle\}\\ \mathcal{B}_1(Q_G) = \emptyset\\ \mathcal{B}_1(Q_S) = \{&\langle [T := 1], \emptyset, \{\neg\mathsf{GVS}(X, 0), \neg\mathsf{ST}(X, \mathsf{us}, 0)\}\rangle,\\ &\langle [T := 0, X := \mathsf{john}], \{\mathsf{BCA}(\mathsf{john}, 1)\}, \emptyset\rangle,\\ &\langle [T := 1], \emptyset, \{\mathsf{BCA}(X, 2)\}\rangle\}\end{aligned}$$

*and we observe that $[T := 0, X := \mathsf{john}]$ is an answer to $Q_S$. In order to perform step 4, we note that the relevant strata are: $\mathsf{GVS}_0$, $\mathsf{GVS}_1$ and $\mathsf{ST}_0$ at the lowest level; $\mathsf{ST}_1$ at the next level; and finally $\mathsf{Risk}_0$ and $\mathsf{Risk}_1$ in the two highest levels, which we omit since no other queries depend on them. We again summarize application of step 4 in a table.*

| $P_t$ | $S(P_t)$ | *set* | *tuples of interest* |
|---|---|---|---|
| $\mathsf{GVS}_0$ | $\emptyset$ | $\mathcal{B}_1(Q_S)$ | $\langle [T := 1], \emptyset, \{\neg\mathsf{GVS}(X,0), \neg\mathsf{ST}(X,\mathsf{us},0)\}\rangle$<br>*becomes*<br>$\langle [T := 1], \{\neg\mathsf{GVS}(X,0)\}, \{\neg\mathsf{ST}(X,\mathsf{us},0)\}\rangle$ |
| $\mathsf{GVS}_1$ | $\emptyset$ | $\mathcal{B}_1(Q_R)$ | $\langle [T := 0], \emptyset, \{\mathsf{BCA}(X,2), \neg\mathsf{GVS}(X,1)\}\rangle$<br>*becomes*<br>$\langle [T := 0], \{\neg\mathsf{GVS}(X,1)\}, \{\mathsf{BCA}(X,2)\}\rangle$ |
| | | $\mathcal{B}_1(Q_R)$ | $\langle [T := 0], \{\neg\mathsf{GVS}(X,0)\}, \{\neg\mathsf{GVS}(X,1), \neg\mathsf{ST}(X,\mathsf{us},0)\}\rangle$<br>*becomes*<br>$\langle [T := 0], \{\neg\mathsf{GVS}(X,0), \neg\mathsf{GVS}(X,1)\}, \{\neg\mathsf{ST}(X,\mathsf{us},0)\}\rangle$ |
| | | $\mathcal{B}_1(Q_R)$ | $\langle [T := 1], \emptyset, \{\neg\mathsf{GVS}(X,2), \neg\mathsf{GVS}(X,1), \neg\mathsf{ST}(X,\mathsf{us},1)\}\rangle$<br>*becomes*<br>$\langle [T := 1], \{\neg\mathsf{GVS}(X,1)\}, \{\neg\mathsf{GVS}(X,2), \neg\mathsf{ST}(X,\mathsf{us},1)\}\rangle$ |
| $\mathsf{ST}_0$ | $[T := 0, X := \mathsf{john}]$<br>$H = \emptyset$ | $\mathcal{B}_1(Q_S)$ | $\langle [T := 1], \{\neg\mathsf{GVS}(X,0)\}, \{\neg\mathsf{ST}(X,\mathsf{us},0)\}\rangle$<br>*becomes, from 4(a),*<br>$\langle [T := 1, X := \mathsf{gus}], \{\neg\mathsf{GVS}(\mathsf{gus},0)\}, \{\neg\mathsf{ST}(\mathsf{gus},\mathsf{us},0)\}\rangle$<br>*and then, from 4(b),*<br>$\langle [T := 1, X := \mathsf{gus}], \{\neg\mathsf{GVS}(\mathsf{gus},0), \neg\mathsf{ST}(\mathsf{gus},\mathsf{us},0)\}, \emptyset\rangle$ |
| | | $\mathcal{B}_1(Q_R)$ | $\langle [T := 0], \{\neg\mathsf{GVS}(X,0), \neg\mathsf{GVS}(X,1)\}, \{\neg\mathsf{ST}(X,\mathsf{us},0)\}\rangle$<br>*becomes, from 4(a),*<br>$\langle [T := 0, X := \mathsf{gus}], \{\neg\mathsf{GVS}(\mathsf{gus},0), \neg\mathsf{GVS}(\mathsf{gus},1)\}, \{\neg\mathsf{ST}(\mathsf{gus},\mathsf{us},0)\}\rangle$<br>*and then, from 4(b),*<br>$\langle [T := 0, X := \mathsf{gus}], \{\neg\mathsf{GVS}(\mathsf{gus},0), \neg\mathsf{GVS}(\mathsf{gus},1), \neg\mathsf{ST}(\mathsf{gus},\mathsf{us},0)\}, \emptyset\rangle$ |
| $\mathsf{ST}_1$ | $[T := 1, X := \mathsf{gus}]$<br>$H = \emptyset$ | $\mathcal{B}_1(Q_R)$ | $\langle [T := 1], \{\neg\mathsf{GVS}(X,1)\}, \{\neg\mathsf{GVS}(X,2), \neg\mathsf{ST}(X,\mathsf{us},1)\}\rangle$<br>*becomes, from 4(a),*<br>$\langle [T := 1, X := \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john},1)\}, \{\neg\mathsf{GVS}(\mathsf{john},2), \neg\mathsf{ST}(\mathsf{john},\mathsf{us},1)\}\rangle$<br>*and then, from 4(b),*<br>$\langle [T := 1, X := \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john},1), \neg\mathsf{ST}(\mathsf{john},\mathsf{us},1)\}, \{\neg\mathsf{GVS}(\mathsf{john},2)\}\rangle$ |

*At the end of this step, we have:*

$$
\begin{aligned}
\mathcal{S}_1(Q_R) = \{&\langle [T := 0], \{\neg\mathsf{GVS}(X,1)\}, \{\mathsf{BCA}(X,2)\}\rangle,\\
&\langle [T := 0, X := \mathsf{gus}], \{\neg\mathsf{GVS}(\mathsf{gus},0), \neg\mathsf{GVS}(\mathsf{gus},1), \neg\mathsf{ST}(\mathsf{gus},\mathsf{us},0)\}, \emptyset\rangle,\\
&\langle [T := 1], \emptyset, \{\mathsf{BCA}(X,3), \neg\mathsf{GVS}(X,2)\}\rangle,\\
&\langle [T := 1, X := \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john},1), \neg\mathsf{ST}(\mathsf{john},\mathsf{us},1)\}, \{\neg\mathsf{GVS}(\mathsf{john},2)\}\rangle\\
\mathcal{S}_1(Q_G) = \emptyset&\\
\mathcal{S}_1(Q_S) = \{&\langle [T := 1, X := \mathsf{gus}], \{\neg\mathsf{GVS}(\mathsf{gus},0), \neg\mathsf{ST}(\mathsf{gus},\mathsf{us},0)\}, \emptyset\rangle,\\
&\langle [T := 0, X := \mathsf{john}], \{\mathsf{BCA}(\mathsf{john},1)\}, \emptyset\rangle,\\
&\langle [T := 1], \emptyset, \{\mathsf{BCA}(X,2)\}\rangle\}
\end{aligned}
$$

*We now consider $\tau = 2$, with $D|_2 = \{\mathsf{GCM}(\mathsf{gus},2), \mathsf{GBOL}(\mathsf{gus},2)\}$. In step 2, the two answers from $\mathcal{S}_1$ with $\mathsf{BCA}(X,2)$ as hypothesis are ignored (since $\mathsf{BCA}(X,2)$ does not unify with any element of*

*$D|_2$) and the remaining ones are copied to $\mathcal{B}_2$. After this step, we have therefore that*

$$\begin{aligned}
\mathcal{B}_2(Q_R) = \{&\langle [T := 0, X := \text{gus}], \{\neg\text{GVS}(\text{gus}, 0), \neg\text{GVS}(\text{gus}, 1), \neg\text{ST}(\text{gus}, \text{us}, 0)\}, \emptyset\rangle,\\
&\langle [T := 1], \emptyset, \{\text{BCA}(X, 3), \neg\text{GVS}(X, 2)\}\rangle,\\
&\langle [T := 1, X := \text{john}], \{\neg\text{GVS}(\text{john}, 1), \neg\text{ST}(\text{john}, \text{us}, 1)\}, \{\neg\text{GVS}(\text{john}, 2)\}\rangle\\
\mathcal{B}_2(Q_G) = \emptyset&\\
\mathcal{B}_2(Q_S) = \{&\langle [T := 1, X := \text{gus}], \{\neg\text{GVS}(\text{gus}, 0), \neg\text{ST}(\text{gus}, \text{us}, 0)\}, \emptyset\rangle,\\
&\langle [T := 0, X := \text{john}], \{\text{BCA}(\text{john}, 1)\}, \emptyset\rangle\}
\end{aligned}$$

*In step 3a we obtain the answer $\langle [T := 2, X := \text{gus}], \{\text{GCM}(\text{gus}, 2), \text{GBOL}(\text{gus}, 2)\}, \emptyset\rangle$ in $\mathcal{B}_2(Q_G)$, while steps 3b and 3c are similar to the previous timestamps. After step 3, we have therefore*

$$\begin{aligned}
\mathcal{B}_2(Q_R) = \{&\langle [T := 0, X := \text{gus}], \{\neg\text{GVS}(\text{gus}, 0), \neg\text{GVS}(\text{gus}, 1), \neg\text{ST}(\text{gus}, \text{us}, 0)\}, \emptyset\rangle,\\
&\langle [T := 1], \emptyset, \{\text{BCA}(X, 3), \neg\text{GVS}(X, 2)\}\rangle,\\
&\langle [T := 1, X := \text{john}], \{\neg\text{GVS}(\text{john}, 1), \neg\text{ST}(\text{john}, \text{us}, 1)\}, \{\neg\text{GVS}(\text{john}, 2)\}\rangle\\
&\langle [T := 2], \emptyset, \{\text{BCA}(X, 4), \neg\text{GVS}(X, 3)\}\rangle,\\
&\langle [T := 2], \emptyset, \{\neg\text{GVS}(X, 3), \neg\text{GVS}(X, 2), \neg\text{ST}(X, \text{us}, 2)\}\rangle\\
\mathcal{B}_2(Q_G) = \{&\langle [T := 2, X := \text{gus}], \{\text{GCM}(\text{gus}, 2), \text{GBOL}(\text{gus}, 2)\}, \emptyset\rangle\}\\
\mathcal{B}_2(Q_S) = \{&\langle [T := 1, X := \text{gus}], \{\neg\text{GVS}(\text{gus}, 0), \neg\text{ST}(\text{gus}, \text{us}, 0)\}, \emptyset\rangle,\\
&\langle [T := 0, X := \text{john}], \{\text{BCA}(\text{john}, 1)\}, \emptyset\rangle,\\
&\langle [T := 2], \emptyset, \{\neg\text{GVS}(X, 1), \neg\text{ST}(X, \text{us}, 1)\}\rangle,\\
&\langle [T := 2], \emptyset, \{\text{BCA}(X, 3)\}\rangle\}
\end{aligned}$$

*The relevant strata for step 4 are now* $\text{GVS}_1$, $\text{GVS}_2$ *and* $\text{ST}_1$ *at the lowest level, followed by* $\text{ST}_2$ *at the next level. As before we can omit* Risk *from the discussion, since no other queries depend on that predicate. This step is summarized in the table below.*

| $P_t$ | $S(P_t)$ | *set* | *tuples of interest* |
|---|---|---|---|
| $\mathsf{GVS}_1$ | $\emptyset$ | $\mathcal{B}_2(Q_S)$ | $\langle [T := 2], \emptyset, \{\neg\mathsf{GVS}(X, 1), \neg\mathsf{ST}(X, \mathsf{us}, 1)\}\rangle$<br>*becomes*<br>$\langle [T := 2], \{\neg\mathsf{GVS}(X, 1)\}, \{\neg\mathsf{ST}(X, \mathsf{us}, 1)\}\rangle$ |
| $\mathsf{GVS}_2$ | $[T := 2, X := \mathsf{gus}]$<br>$H = \emptyset$ | $\mathcal{B}_2(Q_R)$ | $\langle [T := 1], \emptyset, \{\mathsf{BCA}(X, 3), \neg\mathsf{GVS}(X, 2)\}\rangle$<br>*becomes, from 4(a) and 4(b),*<br>$\langle [T := 1, X := \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john}, 2)\}, \{\mathsf{BCA}(\mathsf{john}, 3)\}\rangle$ |
| | | $\mathcal{B}_2(Q_R)$ | $\langle [T := 1, X = \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john}, 1), \neg\mathsf{ST}(\mathsf{john}, \mathsf{us}, 1)\}, \{\neg\mathsf{GVS}(\mathsf{john}, 2)\}\rangle$<br>*becomes*<br>$\langle [T := 1, X = \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john}, 1), \neg\mathsf{ST}(\mathsf{john}, \mathsf{us}, 1), \neg\mathsf{GVS}(\mathsf{john}, 2)\}, \emptyset\rangle$ |
| | | $\mathcal{B}_2(Q_R)$ | $\langle [T := 2], \emptyset, \{\neg\mathsf{GVS}(X, 3), \neg\mathsf{GVS}(X, 2), \neg\mathsf{ST}(X, \mathsf{us}, 2)\}\rangle$<br>*becomes, from 4(a) and 4(b),*<br>$\langle [T := 2, X := \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john}, 2)\}, \{\neg\mathsf{GVS}(\mathsf{john}, 3), \neg\mathsf{ST}(\mathsf{john}, \mathsf{us}, 2)\}\rangle$ |
| $\mathsf{ST}_1$ | $[T := 1, X := \mathsf{gus}]$<br>$H = \emptyset$ | $\mathcal{B}_2(Q_S)$ | $\langle [T := 2], \{\neg\mathsf{GVS}(X, 1)\}, \{\neg\mathsf{ST}(X, \mathsf{us}, 1)\}\rangle$<br>*becomes, from 4(a) and 4(b),*<br>$\langle [T := 2, X := \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john}, 1), \neg\mathsf{ST}(\mathsf{john}, \mathsf{us}, 1)\}, \emptyset\rangle$ |
| $\mathsf{ST}_2$ | $[T := 2, X := \mathsf{john}]$<br>$H = \emptyset$ | $\mathcal{B}_2(Q_R)$ | $\langle [T := 2, X := \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john}, 2)\}, \{\neg\mathsf{GVS}(\mathsf{john}, 3), \neg\mathsf{ST}(\mathsf{john}, \mathsf{us}, 2)\}\rangle$<br>*is removed in step 4(a)* |

*At the end of this step, we have:*

$$\begin{aligned}
\mathcal{S}_2(Q_R) = \{&\langle [T := 0, X := \mathsf{gus}], \{\neg\mathsf{GVS}(\mathsf{gus}, 0), \neg\mathsf{GVS}(\mathsf{gus}, 1), \neg\mathsf{ST}(\mathsf{gus}, \mathsf{us}, 0)\}, \emptyset\rangle,\\
&\langle [T := 1, X := \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john}, 2)\}, \{\mathsf{BCA}(\mathsf{john}, 3)\}\rangle,\\
&\langle [T := 1, X := \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john}, 1), \neg\mathsf{ST}(\mathsf{john}, \mathsf{us}, 1), \neg\mathsf{GVS}(\mathsf{john}, 2)\}, \emptyset\rangle\}\\
&\langle [T := 2], \emptyset, \{\mathsf{BCA}(X, 4), \neg\mathsf{GVS}(X, 3)\}\rangle\\
\mathcal{S}_2(Q_G) = \{&\langle [T := 2, X := \mathsf{gus}], \{\mathsf{GCM}(\mathsf{gus}, 2), \mathsf{GBOL}(\mathsf{gus}, 2)\}, \emptyset\rangle\}\\
\mathcal{S}_2(Q_S) = \{&\langle [T := 1, X := \mathsf{gus}], \{\neg\mathsf{GVS}(\mathsf{gus}, 0), \neg\mathsf{ST}(\mathsf{gus}, \mathsf{us}, 0)\}, \emptyset\rangle,\\
&\langle [T := 0, X := \mathsf{john}], \{\mathsf{BCA}(\mathsf{john}, 1)\}, \emptyset\rangle,\\
&\langle [T := 2, X := \mathsf{john}], \{\neg\mathsf{GVS}(\mathsf{john}, 1), \neg\mathsf{ST}(\mathsf{john}, \mathsf{us}, 1)\}, \emptyset\rangle,\\
&\langle [T := 2], \emptyset, \{\mathsf{BCA}(X, 3)\}\rangle\}
\end{aligned}$$

◁

*Forgetting.* The previous example shows that the online algorithm needs to keep track of a potentially large number of schematic supported answers while running. Furthermore, even answers without premises need to be kept due to the first check in step 3. This may raise concerns about this algorithm having potentially unbounded memory requirements.

For stratified programs, this is not the case, since all schematic answers eventually become answers (with no premises) or are discarded (this follows by combining Proposition 17 with Lemma 19). Furthermore, answers can also be forgotten eventually: static analysis on the sets $\mathcal{P}_Q$ can be used to determine how long an answer can influence the results of the algorithm (essentially, this is a refinement of the notion of the delay of the query), so that all answers can be discarded after enough time has passed. Hypothetical answers can also be discarded when there is an answer with the same substitution (this is the case for the query $Q_R$ and $[T := 1, X := \mathsf{john}]$ at the end of the last example).

*Explicit instantiation.* Step 4 requires instantiating answers with all constants that do not satisfy a particular condition, which can immensely increase the number of schematic answers that have to be considered. In a practical implementation, as mentioned earlier, we expect that this can be addressed by adding constraints to the substitutions in these answers instead of explicitly instantiating them; in our example, in the different applications of step 4(a), we would typically add constraints $X \neq \text{john}$ or $X \neq \text{gus}$ instead of explicitly instantiating $X$. In this manner, we expect that we can partially avoid the exponential blowup that this step of the algorithm might cause.

## 7 RELATED WORK AND DISCUSSION

Our work contributes to the field of stream reasoning, the task of conjunctively reasoning over streaming data and background knowledge [Valle et al. 2009a,b].

Research advances on Complex Event Processors and Data Stream Management Systems [Cugola and Margara 2012], together with Knowledge Representation and the Semantic Web, all contributed to the several stream reasoning languages, systems and mechanisms proposed during the last decade [Dell'Aglio et al. 2017].

Hypothetical query answering over streams can be broadly related to works that study abduction in logic programming [Denecker and Kakas 2002; Inoue 1994], namely those that view negation in logic programs as hypotheses and relate it to contradiction avoidance [Alferes and Pereira 1996; Dung 1991]. 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 [Meadows et al. 2013], but with a rather different approach and in the plan interpretation context. To our knowledge, hypothetical or abductive reasoning has not been previously used in the context of stream reasoning, to answer continuous queries, although it has been applied to annotation of stream data [Alirezaie and Loutfi 2014].

Below we focus on four dimensions of stream reasoning — incremental evaluation, unbound wait and blocking queries, incomplete information, and negation —, which, in our opinion, due to the characteristics of our work, provide the most interesting and meaningful comparison with existing works in the field.

*Incremental evaluation.* 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 [Abiteboul et al. 1995; Barbieri et al. 2010; Gupta et al. 1993; Hu et al. 2018; Motik et al. 2015], on truth maintenance systems [Beck et al. 2015a], window oriented [Ghanem et al. 2007] among others. Our algorithm fits naturally in the first class, as it is an incremental variant of SLD-resolution.

*Unbound wait and blocking queries.* The problems of unbound wait and blocking queries have deserved much attention in the area of query answering over data streams. There have been efforts to identify the problematic issues [Law et al. 2011] and varied proposals to cope with their negative effects, as in [Beck et al. 2015b; Özçep et al. 2014; Ronca et al. 2018a; Zaniolo 2012] among others. Our framework deals with unbound wait by outputting at each time point all supported answers (including some that later may prove to be false), as illustrated e.g. in Example 10. Furthermore, if we receive a supported answer whose time parameters are all instantiated, then we immediately have a bound on how long we have to wait until the answer is definite (or rejected).

Blocking queries may still be a problem in our framework, though: as seen in Example 7, blocking operations (in the form of infinitely recursive predicates) may lead to infinite SLD-derivations, which cause the pre-processing step of our algorithm to diverge. We showed that syntactic restrictions

of the kind already considered by [Ronca et al. 2018b] are guaranteed to avoid blocking queries; Example 8 illustrates that, as in the cited work, these conditions are however not necessary: pre-processing may terminate even though they do not hold.

Our algorithm also implicitly embodies a forgetting algorithm, as the only elements of $D_\tau$ kept in the $E$ sets are those that are still relevant to compute future answers to $Q$.

The approaches from [Ronca et al. 2018a; Walega et al. 2019; Zaniolo 2012], which are also specifically Datalog-based, make some restrictions to the form of rules in order to avoid blocking behaviours.

In [Zaniolo 2012], working in a variant of Datalog called Streamlog, the author characterises *sequential rules and programs* with the purpose of avoiding blocking behaviour. In particular, the rule $\mathsf{Shdn}(X,T) \rightarrow \mathsf{Malf}(X,T-2)$ from Example 1 is disallowed in this framework, since the timestamp in the head of a rule may never be smaller than the timestamps of the atoms in the body.

The authors of [Ronca et al. 2018a] propose the *language of forward-propagating queries*, yet another variant of Temporal Datalog that allows queries to be answered in polynomial time in the size of the input data. This is again achieved at the cost of disallowing propagation of derived facts towards past time points – once more precluding rule $\mathsf{Shdn}(X,T) \rightarrow \mathsf{Malf}(X,T-2)$ in Example 1. The authors present a generic algorithm to compute answers to a query, given a window size, and investigate methods for calculating a minimal window size.

Memory consumption is a concern in [Walega et al. 2019], where a sound and complete stream reasoning algorithm is presented for a fragment of datalogMTL – forward-propagating datalogMTL – that also disallows propagation of derived information towards the past. DatalogMTL is a Datalog extension of the Horn fragment of MTL [Alur and Henzinger 1993; Koymans 1990], which was proposed in [Brandt et al. 2018] for ontology-based access to temporal log data. DatalogMTL rules allow propagation of derived information to both past and future time points. Concerned with the possibly huge or unbounded set of input facts that have to be kept in memory, the authors of [Walega et al. 2019] 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.

More recently, [Ronca et al. 2022] removes these syntactic restrictions on the forms of rules (keeping only the light requirement of temporal guardedness, which is similar to connectedness) and studies fragments of the language for which the existence of a query delay is decidable.

*Incomplete information.* When reasoning over sources with incomplete information, the concepts of certain and possible answers, and granularities thereof, inevitably arise [de Leng and Heintz 2019; Gray et al. 2007; Lang et al. 2014; Razniewski et al. 2015] as a way to assign confidence levels to the information output to the user. These approaches, like ours, also compute answers with incomplete information.

However, our proposal is substantially different from those works, since they focus on answers over past incomplete information. First, as mentioned in Section 2, we assume that our time stream is complete, in the sense that whenever it produces a fact about a time instant $\tau$, all EDB facts about time instants $\tau' < \tau$ are already there (in line with the progressive closed world assumption of [Zaniolo 2012]). Secondly, our hypothetical and supported answers are built over present facts and future, still unknown, hypotheses, and eventually either become effective answers or are discarded; in the scenario of past incomplete information, the confidence level of answers may never change.

*Negation.* Other authors have considered languages using negation. Our definition of stratification is similar to the concept of temporal stratification from [Zaniolo 2015]. However, temporal

stratification requires the strata to be also ordered according to time (i.e., if the definition of $p_t$ is in $\Pi_i$ and the definition of $p_{t+1}$ is in $\Pi_j$, then $j \geq i$). We make no such assumption in this work.

Beck et al. [Beck et al. 2015a] also consider a notion of temporal stratification for stream reasoning. However, their framework also includes explicit temporal operators, making the whole formalization more complex.

Insofar as we are aware, this is also the first framework for continuous query answering that does not require negation to be safe.

*Practical applicability.* The complexity bounds for the different steps of the algorithm, and the potentially large sizes of the sets of schematic hypothetical answers that have to be kept during execution, raise the question of whether this framework has practical applicability.

From the theoretical point of view, we point out that the bounds given (e.g. in Theorem 8) are over-approximations based on a worst-case analysis. Furthermore, even in that result, the exponential dependency is on only one parameter (the highest arity of a predicate in the program), which for handwritten programs will likely tend to be low.

Another observation concerns the existence of a query delay, which always exists for connected and nonrecursive queries. If there is a delay for the query, then the size of the set of schematic hypothetical answers cannot grow unboundedly, since unresolved answers will unavoidably be automatically discarded when the delay is reached.

In a separated work [Cruz-Filipe et al. 2023], some of the authors have applied this formalism to a concrete problem with interest for the media industry: trying to predict leading news topics based on real-time data from Google Trends. This was achieved by writing a program with rules that capture when a topic is considered to be trending, and dynamically computing hypothetical answers to the query "is topic $X$ trending?". In this experiment, the datastream produces data once per hour. The corresponding update time needed is in the order of magnitude of a few seconds, making it completely unproblematic. The data that needs to be stored (not reported in the article) is also very modest in size, and poses no practical problems.

That experiment does not make use of negation. Although the algorithms for programs including negations have a higher complexity, we have discussed in the corresponding section how practical implementations can adopt strategies to mitigate these effects.

For these reasons, we are optimistic and believe that there is practical potential in our framework. We plan to investigate this further by applying it to some more real-world scenarios, preferably in concrete cases provided by industry.

## 8 CONCLUSIONS AND FUTURE WORK

We introduced a formalism for dealing with answers to continuous queries that are consistent with the information that is available, but may still need further confirmation from future data. We defined the notions of hypothetical and supported answers declaratively, and showed how they can be computed by means of a variant of SLD-resolution. By refining this idea, we designed a two-phase algorithm whose online component maintains and updates the set of supported answers. Our framework also allows for negation, whose declarative semantics is in line with the progressive closed world assumption.

Our methodology avoids some of the typical problems with delay in continuous query answering. In particular, hypothetical answers allow us to detect that we are not in a situation of unbound wait: if we receive a supported answer whose time parameters are all instantiated, then we immediately have a bound on how long we have to wait until the answer is definite (or rejected). The usefulness of this information is of course application specific. In the situation of Example 9, we could for example take extra preventive measures to ensure that $\mathsf{Temp}(\mathsf{wt25}, \mathsf{high}, 2)$ does not become true.

The offline step of our two-phase algorithm may diverge due to issues related to blocking queries. We showed that adequate syntactic restrictions prevent this situation from arising, but that they are not necessary.

For nonrecursive and connected queries without negation, the online step of our algorithm runs in polynomial time. However, it may still be computationally heavy if the sets of hypothetical answers that it needs to compute are large. In particular, pre-processing may produce output that is exponential in the size of the original program. We minimize the impact of this by representing hypothetical answers schematically (since we keep variables uninstantiated), thus limiting the space and time requirements for this algorithm. By adding instantiation constraints to this representation we can also reduce the impact of the exponential blowup introduced by negation. These are informal observations though, which we plan to measure more precisely in a future practical evaluation.

The sets of evidence for hypothetical answers can be used to define a partial order of relative "confidence" of each hypothetical answer. It would be interesting to explore the connection between such notions of confidence and other frameworks of reasoning with multiple truth values.

We also plan to implement our algorithm and evaluate its performance in practice – a first step in this direction, only for the positive fragment of the language, has already been undertaken. It would also be interesting to explore approximation techniques for online algorithms in order to allow for unsafe forgetting, with bounds on how many wrong answers may be produced, and to extend our formalism to allow for communication delays, which are present in many interesting practical scenarios.

## ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers, whose suggestions contributed to improving this article.

This work was partially supported by the Independent Research Fund Denmark, grant DFF-7014-00041, and by FCT through the LASIGE Research Unit, ref. UIDB/00408/2020 and ref. UIDP/00408/2020.

## REFERENCES

AAAI2019 2019. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA*. AAAI Press.

Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases.* Addison-Wesley.

José Júlio Alferes and Luís Moniz Pereira. 1996. *Reasoning with Logic Programming*. Lecture Notes in Computer Science, Vol. 1111. Springer.

Marjan Alirezaie and Amy Loutfi. 2014. Automated Reasoning using Abduction for Interpretation of Medical Signals. *J. Biomed. Semant.* 5 (2014), 35.

Rajeev Alur and Thomas A. Henzinger. 1993. Real-Time Logics: Complexity and Expressiveness. *Inf. Comput.* 104, 1 (1993), 35–77.

Arvind Arasu, Shivnath Babu, and Jennifer Widom. 2006. The CQL continuous query language: semantic foundations and query execution. *VLDB J.* 15, 2 (2006), 121–142.

Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. 2002. Models and Issues in Data Stream Systems. In *21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 2002, Madison, Wisconsin, USA*, Lucian Popa, Serge Abiteboul, and Phokion G. Kolaitis (Eds.). ACM, 1–16.

Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. 2009. C-SPARQL: SPARQL for continuous querying. In *18th International Conference on World Wide Web, WWW 2009, Madrid, Spain*, Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl (Eds.). ACM, 1061–1062.

Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. 2010. Incremental Reasoning on Streams and Rich Background Knowledge. In *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Greece, Part I (LNCS, Vol. 6088)*, Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache (Eds.). Springer, 1–15.

Harald Beck, Minh Dao-Tran, and Thomas Eiter. 2015a. Answer Update for Rule-Based Stream Reasoning. In *34th International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina*, Qiang Yang and Michael J. Wooldridge (Eds.). AAAI Press, 2741–2747.

Harald Beck, Minh Dao-Tran, Thomas Eiter, and Michael Fink. 2015b. LARS: A Logic-Based Framework for Analyzing Reasoning over Streams, See [Bonet and Koenig 2015], 1431–1438.

Mordechai Ben-Ari. 2012. *Mathematical Logic for Computer Science* (3rd ed.). Springer.

Blai Bonet and Sven Koenig (Eds.). 2015. *29th AAAI Conference on Artificial Intelligence, AAAI 2015, Austin, TX, USA*. AAAI Press.

Sebastian Brandt, Elem Güzel Kalayci, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. 2018. Querying Log Data with Metric Temporal Logic. *J. Artif. Intell. Res.* 62 (2018), 829–877.

Jan Chomicki and Tomasz Imielinski. 1988. Temporal Deductive Databases and Infinite Objects. In *7th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 1988, Austin, Texas, USA*, Chris Edmondson-Yurkanan and Mihalis Yannakakis (Eds.). ACM, 61–73.

Luís Cruz-Filipe, Graça Gaspar, and Isabel Nunes. 2020. Hypothetical answers to continuous queries over data streams. In *34th AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA*. AAAI Press, 2798–2805.

Luís Cruz-Filipe, Sofia Kostopoulou, Fabrizio Montesi, and Jonas Vistrup. 2023. $\mu$XL: Explainable Lead Generation with Microservices and Hypothetical Answers. (2023). accepted for publication.

Gianpaolo Cugola and Alessandro Margara. 2012. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.* 44, 3 (2012), 15:1–15:62.

Minh Dao-Tran and Thomas Eiter. 2017. Streaming Multi-Context Systems. In *26th International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia*, Carles Sierra (Ed.). ijcai.org, 1000–1007.

Daniel de Leng and Fredrik Heintz. 2019. Approximate Stream Reasoning with Metric Temporal Logic under Uncertainty, See [AAAI2019 2019], 2760–2767.

Daniele Dell'Aglio, Emanuele Della Valle, Frank van Harmelen, and Abraham Bernstein. 2017. Stream reasoning: A survey and outlook. *Data Sci.* 1, 1–2 (2017), 59–83.

Marc Denecker and Antonis C. Kakas. 2002. Abduction in Logic Programming. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I (Lecture Notes in Computer Science, Vol. 2407)*, Antonis C. Kakas and Fariba Sadri (Eds.). Springer, 402–436.

Phan Minh Dung. 1991. Negations as Hypotheses: An Abductive Foundation for Logic Programming. In *Logic Programming, Proceedings of the Eigth International Conference, Paris, France, June 24-28, 1991*, Koichi Furukawa (Ed.). MIT Press, 3–17.

Michael Gelfond and Vladimir Lifschitz. 1988. The Stable Model Semantics for Logic Programming, See [Kowalski and Bowen 1988], 1070–1080.

Thanaa M. Ghanem, Moustafa A. Hammad, Mohamed F. Mokbel, Walid G. Aref, and Ahmed K. Elmagarmid. 2007. Incremental Evaluation of Sliding-Window Queries over Data Streams. *IEEE Trans. Knowl. Data Eng.* 19, 1 (2007), 57–72.

Alasdair J.G. Gray, Werner Nutt, and M. Howard Williams. 2007. Answering queries over incomplete data stream histories. *IJWIS* 3, 1/2 (2007), 41–60.

Ashish Gupta, Inderpal Singh Mumick, and V.S. Subrahmanian. 1993. Maintaining Views Incrementally. In *ACM SIGMOD International Conference on Management of Data, SIGMOD 1993, Washington, DC, USA*, Peter Buneman and Sushil Jajodia (Eds.). ACM Press, 157–166.

Pan Hu, Boris Motik, and Ian Horrocks. 2018. Optimised Maintenance of Datalog Materialisations, See [McIlraith and Weinberger 2018], 1871–1879.

Katsumi Inoue. 1994. Hypothetical Reasoning in Logic Programs. *J. Log. Program.* 18, 3 (1994), 191–227.

Phokion G. Kolaitis. 1991. The Expressive Power of Stratified Programs. *Inf. Comput.* 90, 1 (1991), 50–66.

Robert A. Kowalski and Kenneth A. Bowen (Eds.). 1988. *5th International Conference and Symposium on Logic Programming, ICLP 1988, Seattle, Washington, USA*. MIT Press.

Ron Koymans. 1990. Specifying Real-Time Properties with Metric Temporal Logic. *Real-Time Systems* 2, 4 (1990), 255–299.

Willis Lang, Rimma V. Nehme, Eric Robinson, and Jeffrey F. Naughton. 2014. Partial results in database systems. In *ACM SIGMOD International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA*, Curtis E. Dyreson, Feifei Li, and M. Tamer Özsu (Eds.). ACM, 1275–1286.

Yan-Nei Law, Haixun Wang, and Carlo Zaniolo. 2011. Relational languages and data models for continuous queries on sequences and data streams. *ACM Trans. Database Syst.* 36, 2 (2011), 8:1–8:32.

John W. Lloyd. 1984. *Foundations of Logic Programming*. Springer.

Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). 2018. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, LA, USA*. AAAI Press.

Ben Leon Meadows, Pat Langley, and Miranda Jane Emery. 2013. Seeing Beyond Shadows: Incremental Abductive Reasoning for Plan Understanding. In *Plan, Activity, and Intent Recognition, Papers from the 2013 AAAI Workshop, Bellevue, Washington, USA, July 15, 2013 (AAAI Workshops, Vol. WS-13-13)*. AAAI.

Boris Motik, Yavor Nenov, Robert Edgar Felix Piro, and Ian Horrocks. 2015. Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm, See [Bonet and Koenig 2015], 1560–1568.

Özgür Lütfü Özçep, Ralf Möller, and Christian Neuenstadt. 2014. A Stream-Temporal Query Language for Ontology Based Data Access. In *KI 2014: Advances in Artificial Intelligence – 37th Annual German Conference on AI, Stuttgart, Germany (LNCS, Vol. 8736)*, Carsten Lutz and Michael Thielscher (Eds.). Springer, 183–194.

Luigi Palopoli. 1992. Testing Logic Programs for Local Stratification. *Theor. Comput. Sci.* 103, 2 (1992), 205–234.

Teodor C. Przymusinski. 1988a. On the Declarative Semantics of Deductive Databases and Logic Programs. In *Foundations of Deductive Databases and Logic Programming*, Jack Minker (Ed.). Morgan Kaufmann, 193–216.

Teodor C. Przymusinski. 1988b. Perfect Model Semantics, See [Kowalski and Bowen 1988], 1081–1096.

Simon Razniewski, Flip Korn, Werner Nutt, and Divesh Srivastava. 2015. Identifying the Extent of Completeness of Query Answers over Partially Complete Databases. In *ACM SIGMOD International Conference on Management of Data, SIGMOD 2015, Melbourne, Victoria, Australia*, Timos K. Sellis, Susan B. Davidson, and Zachary G. Ives (Eds.). ACM, 561–576.

Alessandro Ronca, Mark Kaminski, Bernardo Cuenca Grau, and Ian Horrocks. 2018a. The Window Validity Problem in Rule-Based Stream Reasoning. In *16th International Conference on Principles of Knowledge Representation and Reasoning, KR 2018, Tempe, AZ, USA*, Michael Thielscher, Francesca Toni, and Frank Wolter (Eds.). AAAI Press, 571–581.

Alessandro Ronca, Mark Kaminski, Bernardo Cuenca Grau, and Ian Horrocks. 2022. The delay and window size problems in rule-based stream reasoning. *Artif. Intell.* 306 (2022), 103668. https://doi.org/10.1016/j.artint.2022.103668

Alessandro Ronca, Mark Kaminski, Bernardo Cuenca Grau, Boris Motik, and Ian Horrocks. 2018b. Stream Reasoning in Temporal Datalog, See [McIlraith and Weinberger 2018], 1941–1948.

Michael Stonebraker, Ugur Çetintemel, and Stanley B. Zdonik. 2005. The 8 requirements of real-time stream processing. *SIGMOD Record* 34, 4 (2005), 42–47.

Emanuele Della Valle, Stefano Ceri, Davide Francesco Barbieri, Daniele Braga, and Alessandro Campi. 2009a. A First Step Towards Stream Reasoning. In *Future Internet – FIS 2008, Vienna, Austria (LNCS, Vol. 5468)*, John Domingue, Dieter Fensel, and Paolo Traverso (Eds.). Springer, 72–81.

Emanuele Della Valle, Stefano Ceri, Frank van Harmelen, and Dieter Fensel. 2009b. It's a Streaming World! Reasoning upon Rapidly Changing Information. *IEEE Intelligent Systems* 24, 6 (2009), 83–89.

Przemyslaw Andrzej Walega, Mark Kaminski, and Bernardo Cuenca Grau. 2019. Reasoning over Streaming Data in Metric Temporal Datalog, See [AAAI2019 2019], 3092–3099.

Carlo Zaniolo. 2012. Logical Foundations of Continuous Query Languages for Data Streams. In *Datalog in Academia and Industry – 2nd International Workshop, Datalog 2.0, Vienna, Austria (LNCS, Vol. 7494)*, Pablo Barceló and Reinhard Pichler (Eds.). Springer, 177–189.

Carlo Zaniolo. 2015. Expressing and Supporting Efficiently Greedy Algorithms as Locally Stratified Logic Programs. In *Technical Communications of the 31st International Conference and Symposium on Logic Programming, ICLP 2015, Cork, Ireland (CEUR Workshop Proceedings, Vol. 1433)*, Marina De Vos, Thomas Eiter, Yuliya Lierler, and Francesca Toni (Eds.). CEUR-WS.org.