Stratified Negation in Datalog with Metric Temporal Operators

David J. Tena Cucala,[1] Przemysław A. Wałęga,[1] Bernardo Cuenca Grau,[1] Egor V. Kostylev[2]
[1]Department of Computer Science, University of Oxford
[2]Department of Informatics, University of Oslo
{david.tena.cucala, przemyslaw.walega, bernardo.cuenca.grau}@cs.ox.ac.uk, egork@ifi.uio.no

Abstract

We extend DatalogMTL—Datalog with operators from metric temporal logic—by adding stratified negation as failure. The new language provides additional expressive power for representing and reasoning about temporal data and knowledge in a wide range of applications. We consider models over the rational timeline, study their properties, and establish the computational complexity of reasoning. We show that, as in negation-free DatalogMTL, fact entailment in our language is PSPACE-complete in data and EXPSPACE-complete in combined complexity. Thus, the extension with stratified negation does not lead to higher complexity.

Introduction

We consider DatalogMTL (Brandt et al. 2017, 2018; Wałęga et al. 2019)—a temporal extension of Datalog, where atoms in rules can mention operators from metric temporal logic MTL (Koymans 1990) interpreted over the rational timeline. For example, the metric expression $\diamondminus_{(0,10s]} Sym(x, \mathsf{cough})$ states that x coughed at least once in the last 10 seconds, whereas $\boxminus_{(0,10s]} Sym(x, \mathsf{cough})$ states that x coughed continuously in the described period. By allowing such expressions in rules, DatalogMTL provides a powerful language for representing and reasoning about information involving temporal intervals, which is expressive enough to capture other well-known temporal rule languages such as Datalog$_{1S}$ (Chomicki and Imieliński 1988, 1989) and *Templog* (Abadi and Manna 1989). As a result, DatalogMTL has been suggested as a suitable formalism for applications such as temporal ontology-based data access (Brandt et al. 2018), stream reasoning (Wałęga, Cuenca Grau, and Kaminski 2019), and temporal logic programming (Brzoska 1998).

As a running example, let us consider the formalisation of some of the COVID-19 self-isolation rules for households in the United Kingdom[1] using DatalogMTL:

$$Sym(x, \mathsf{cont\_cough}) \leftarrow \boxminus_{[0,5m]} \diamondminus_{(0,10s]} Sym(x, \mathsf{cough}),$$
$$COVIDSym(x) \leftarrow \diamondminus_{[0,24h]} Sym(x, \mathsf{cont\_cough}),$$
$$COVIDSym(x) \leftarrow \diamondminus_{[0,24h]} Sym(x, \mathsf{fever}).$$

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]https://www.gov.uk/government/publications/covid-19-stay-at-home-guidance/stay-at-home-guidance-for-households-with-possible-coronavirus-covid-19-infection

The first rule defines the continuous cough symptom as a cough that has occurred at least once every 10 seconds (expressed using the operator $\diamondminus_{(0,10s]}$) within the previous 5 minutes (expressed using $\boxminus_{[0,5m]}$); the remaining rules establish that anyone with a continuous cough or a fever in the last 24 hours is displaying COVID-19 symptoms.

An important limitation of DatalogMTL as defined in prior work is that it does not allow negation in rules. Non-monotonic negation applied to temporal information is, however, a very useful feature that has been recently considered in the contexts of temporal answer set programming over the integer timeline (Cabalar et al. 2019, 2020) and stream reasoning (Beck, Dao-Tran, and Eiter 2018; Zaniolo 2012; Das, Gandhi, and Zaniolo 2018).

In this paper, we propose and study the language of DatalogMTL with stratified negation as failure. Our language extends positive (i.e., negation-free) DatalogMTL (over the rational timeline and under continuous semantics) as well as plain Datalog with stratified negation. It provides useful additional expressivity for a wide range of application scenarios. For instance, consider the guidance that people with COVID-19 symptoms must remain in home isolation for 10 days after the onset of their symptoms. We can attempt to represent this policy using the following rule:

$$\boxplus_{[0,10d]} Isol(x) \leftarrow COVIDSym(x).$$

Here, the future operator $\boxplus_{[0,10d]}$ states that isolation should continue uninterrupted for 10 days after symptoms are detected. This rule, however, does not faithfully represent the aforementioned policy; in particular, it implies that a patient must isolate for 10 days in future *for as long as they display symptoms*, where the UK's guidance dictates that the 10 day self-isolation is effective *from the onset of symptoms*; indeed, a patient can stop isolating after that period since symptoms like a continuous cough can linger for several weeks after the infection has gone. Negation as failure, which is denoted by not, can be used to faithfully capture the requirement that symptoms must be new for an isolation period to start; in particular, the following rule states that an isolation period is triggered whenever a patient first displays symptoms after a period of 10 days without showing signs of the disease:

$$\boxplus_{[0,10d]} Isol(x) \leftarrow COVIDSym(x) \land \mathsf{not}\ \diamondminus_{(0,10d]} COVIDSym(x).$$

Our contributions in this paper are as follows.

First, we define the syntax and semantics of DatalogMTL programs with stratified negation. Our semantics is a natural extension of the semantics of both Datalog with stratified negation and positive DatalogMTL. Fact entailment for a program and dataset is decided using a unique interpretation, called the *materialisation*, defined as the last element of a sequence of interpretations, called *partial materialisations*, starting with the least model of the data and subsequently extending them to satisfy higher strata of the program.

In contrast to Datalog with stratified negation, however, we cannot check fact entailment by simply constructing the sequence of partial materialisations since each of them can now be of infinite size. To overcome this difficulty, we define Büchi automata allowing us to check if a particular fact holds in a partial materialisation without having to construct the partial materialisation entirely. This, in turn, allows us to establish tight complexity bounds for fact entailment, namely a PSPACE bound in data complexity and an EXPSPACE bound in combined complexity; thus, fact entailment in our language is not harder than in positive DatalogMTL (Brandt et al. 2018; Wałęga et al. 2019) and the additional expressivity comes at no computational cost. Our upper bounds transfer also to Datalog$_{1S}$ with stratified negation.

Preliminaries

Time and Intervals. The *(rational) timeline* is the (ordered) set $\mathbb{Q}$ of rational numbers, and a *time point* is an element of the timeline. An *interval*, ϱ, is a subset of $\mathbb{Q}$ such that for all $t_1, t_2, t_3 \in \mathbb{Q}$ satisfying $t_1 < t_2 < t_3$ and $t_1, t_3 \in \varrho$ we have $t_2 \in \varrho$, if ϱ is bounded below then it has a greatest lower bound in $\mathbb{Q}$, and if ϱ is bounded above then it has a least upper bound in $\mathbb{Q}$. The *left endpoint* ϱ^- of ϱ is the largest rational lower bound of ϱ, if one exists, or $-\infty$ otherwise; the *right endpoint* ϱ^+ of ϱ is the smallest rational upper bound of ϱ if one exists, or ∞ otherwise. An interval is *punctual* if it contains exactly one number; it is *positive* if it does not contain negative numbers; and it is *bounded* if both its left and right endpoints are rational numbers.

We use the standard *bracket representation* $\langle \varrho^-, \varrho^+ \rangle$ for an interval ϱ, where the left bracket $\langle$ is either $[$ or $($ and the right bracket $\rangle$ is either $]$ or $)$. Brackets $[$ and $]$ indicate that the corresponding endpoints are included in the interval, whereas $($ and $)$ indicate that they are excluded. We use the generic symbols $\langle$ and $\rangle$ if brackets are not determined, and we write a punctual interval $[t, t]$ as t. Rational endpoints are written as (not necessary reduced) fractions with integer numerators and positive integer denominators, both in binary.

Metric Atoms and Interpretations. We consider a signature consisting of (disjoint) sets of constants and predicates. A *relational atom* is an expression $P(\mathbf{s})$ where P is a predicate and $\mathbf{s}$ is a tuple of constants and variables of length matching the arity of P. A *metric atom* M is an expression specified by the following grammar, where $P(\mathbf{s})$ ranges over relational atoms and ϱ over positive intervals:

$$M ::= \top \mid \bot \mid P(\mathbf{s}) \mid \diamondminus_\varrho M \mid \diamondplus_\varrho M \mid \boxminus_\varrho M \mid \boxplus_\varrho M \mid M \mathcal{S}_\varrho M \mid M \mathcal{U}_\varrho M.$$

$\mathfrak{I}, t \models \top$	for each $t \in \mathbb{Q}$
$\mathfrak{I}, t \models \bot$	for no $t \in \mathbb{Q}$
$\mathfrak{I}, t \models \diamondminus_\varrho M$	iff $\mathfrak{I}, t' \models M$ for some t' with $t - t' \in \varrho$
$\mathfrak{I}, t \models \diamondplus_\varrho M$	iff $\mathfrak{I}, t' \models M$ for some t' with $t' - t \in \varrho$
$\mathfrak{I}, t \models \boxminus_\varrho M$	iff $\mathfrak{I}, t' \models M$ for all t' with $t - t' \in \varrho$
$\mathfrak{I}, t \models \boxplus_\varrho M$	iff $\mathfrak{I}, t' \models M$ for all t' with $t' - t \in \varrho$
$\mathfrak{I}, t \models M_1 \mathcal{S}_\varrho M_2$	iff $\mathfrak{I}, t' \models M_2$ for some t' with $t - t' \in \varrho$ and $\mathfrak{I}, t'' \models M_1$ for all $t'' \in (t', t)$
$\mathfrak{I}, t \models M_1 \mathcal{U}_\varrho M_2$	iff $\mathfrak{I}, t' \models M_2$ for some t' with $t' - t \in \varrho$ and $\mathfrak{I}, t'' \models M_1$ for all $t'' \in (t, t')$

Table 1: Semantics of ground metric atoms

A metric atom is *ground* if it mentions no variables. A *metric fact* is an expression $M@\varrho$, with M a ground metric atom and ϱ a non-empty interval; it is *relational* if so is M. A *dataset* is a finite set of relational facts.

An *interpretation* $\mathfrak{I}$ specifies, for each ground relational atom $P(\mathbf{c})$ and each time point $t \in \mathbb{Q}$, whether $P(\mathbf{c})$ is *satisfied* at t, in which case we write $\mathfrak{I}, t \models P(\mathbf{c})$. This notion extends to other ground metric atoms as given in Table 1. An interpretation $\mathfrak{I}$ is a *model* of a metric fact $M@\varrho$, written $\mathfrak{I} \models M@\varrho$, if $\mathfrak{I}, t \models M$ for all $t \in \varrho$; and it is a *model* of a set $\mathfrak{M}$ of metric facts (e.g., a dataset) if it is a model of all facts in $\mathfrak{M}$. An interpretation $\mathfrak{I}$ *contains* an interpretation $\mathfrak{I}'$, written $\mathfrak{I}' \subseteq \mathfrak{I}$, if $\mathfrak{I}', t \models P(\mathbf{c})$ implies $\mathfrak{I}, t \models P(\mathbf{c})$, for each ground relational atom $P(\mathbf{c})$ and time point $t \in \mathbb{Q}$. Furthermore, $\mathfrak{I}$ is the *least* interpretation in a set X of interpretations, if $\mathfrak{I} \subseteq \mathfrak{I}'$ for every $\mathfrak{I}' \in X$. Note that we use the *continuous semantics* over the rationals (Brandt et al. 2018), but DatalogMTL has also been studied under the *pointwise semantics* (Ryzhikov, Wałęga, and Zakharyaschev 2019, 2020) and over the integer timeline (Wałęga et al. 2020).

DatalogMTL with Stratified Negation

In this section we introduce the syntax and semantics of DatalogMTL$^\neg$, which extends DatalogMTL by adding stratified negation as failure.

The syntax of DatalogMTL$^\neg$ is defined analogously to standard Datalog with negation (Abiteboul, Hull, and Vianu 1995; Dantsin et al. 2001): the rule body is a conjunction of atoms and negated atoms, and the rule head is a single atom distinct from $\bot$. The only difference is that, as in DatalogMTL (Brandt et al. 2017), we allow metric operators in atoms and disallow $\diamondminus$, $\diamondplus$, $\mathcal{S}$, and $\mathcal{U}$ in rule heads.

Definition 1. *A* rule *is an expression r of the form*

$$M \leftarrow M_1 \wedge \cdots \wedge M_k \wedge \mathsf{not}\, M_{k+1} \wedge \cdots \wedge \mathsf{not}\, M_{k+m}, \quad (1)$$

where $k, m \geq 0$, each M_i is a metric atom, and M is a metric atom specified by the following grammar, where $P(\mathbf{s})$ ranges over relational atoms and ϱ over positive intervals:

$$M ::= P(\mathbf{s}) \mid \boxminus_\varrho M \mid \boxplus_\varrho M.$$

The metric atom M in Form (1) *is the* head *of r, while the conjunction therein is the* body *of r; the atoms $M_1, \ldots, M_k$ are the* positive body atoms *of r, and $M_{k+1}, \ldots, M_{k+m}$ are its* negated body atoms. *A rule r is* safe *if each variable it mentions occurs in some positive body atom, it is* ground *if it has no variables, and it is* positive *if it has no negated body atoms. A (*DatalogMTL$^\neg$*)* program *is a finite set of safe rules; it is* ground *or* positive *if all its rules are.*

The semantics of Datalog programs with negation is not straightforward and there have been many proposals over the years. There is, however, a general consensus on the semantics of *stratifiable programs*, which can be organised in layers of sub-programs, called *strata*, so that, for each rule in a stratum, every predicate appearing in a positive body atom does not appear in the heads in the strata above and every predicate appearing in a negated atom does not appear in the heads either in the same stratum or in the strata above. We adopt an analogous semantics for DatalogMTL$^\neg$, and so, we can reuse the well-known techniques to check in polynomial time whether a program is stratifiable and to compute a corresponding stratification (Dantsin et al. 2001). We note, however, that alternative definitions of stratifiable programs in temporal logics have been considered which are based, for example, on the notions of local and temporal stratifications (Przymusiński 1988; Nomikos, Rondogiannis, and Gergatsoulis 2005; Koutras and Nomikos 2000).

Definition 2. *A* stratification *of a program Π is a function σ mapping predicates mentioned in Π to positive integers such that the following holds, for each rule $r \in \Pi$, and all predicates P, P^+ and P^- mentioned in the head, positive body atoms and negated body atoms of r, respectively:*

$$\sigma(P^+) \leq \sigma(P) \qquad \textit{and} \qquad \sigma(P^-) < \sigma(P).$$

A program is stratifiable *if it has a stratification.*

For each $s \in \mathbb{N}$, the s-th stratum *Π_s^σ of Π with respect to a stratification σ is the (possibly empty) subset of Π containing each rule with $\sigma(P) = s$ for the predicate P in the head. We also let $\Pi_{\leq s}^\sigma = \Pi_1^\sigma \cup \cdots \cup \Pi_s^\sigma$.*

In what follows, we define the semantics of stratifiable programs, which generalises not only the semantics of Datalog with stratified negation, but also of DatalogMTL. First, we extend the semantics of ground metric atoms from Table 1 to capture the meaning of negated atoms, as well as define models of DatalogMTL$^\neg$ programs.

Definition 3. *For every interpretation $\mathfrak{I}$, time point $t \in \mathbb{Q}$, and ground metric atom M, the interpretation $\mathfrak{I}$* satisfies $\mathsf{not}\, M$ *at t, denoted by $\mathfrak{I}, t \models \mathsf{not}\, M$, if $\mathfrak{I}, t \not\models M$. An interpretation $\mathfrak{I}$ is a* model *of a rule r of Form* (1) *if, for every assignment ν of variables to constants making r ground and for every $t \in \mathbb{Q}$, we have $\mathfrak{I}, t \models \nu(M)$ whenever $\mathfrak{I}, t \models \nu(M_i)$ for each $i \in \{1, \ldots, k\}$ and $\mathfrak{I}, t \models \mathsf{not}\, \nu(M_i)$ for each $i \in \{k+1, \ldots, k+m\}$. An interpretation $\mathfrak{I}$ is a* model *of a program Π if $\mathfrak{I}$ is a model of each rule in Π.*

For each stratifiable program and dataset, we define a unique model, which we call their *materialisation*. We define this model as in Datalog: given a stratification, we introduce a finite sequence of interpretations starting with the least model of the dataset and subsequently extending, in a minimal way, the valuation of predicates defined by a given stratum (i.e., predicates P such that $\sigma(P) = s$ for the stratum number s) to satisfy this stratum. The existence of such extensions follows from the following proposition.

Proposition 4. *Let Π be a program with a stratification σ, let $s \geq 1$, and let $\mathfrak{I}$ be an interpretation. Then, there exists a unique least interpretation amongst the set of interpretations $\mathfrak{I}'$ satisfying the following:*

- *$\mathfrak{I} \subseteq \mathfrak{I}'$;*
- *$\mathfrak{I}'$ agrees with $\mathfrak{I}$ on all predicates P with $\sigma(P) \neq s$; and*
- *$\mathfrak{I}'$ is a model of Π_s^σ.*

Proof sketch. The required interpretation can be obtained by extending $\mathfrak{I}$ with only those facts over predicates P with $\sigma(P) = s$ that are necessary to satisfy Π_s^σ. To show that there is a unique such an interpretation, recall that $\boxminus$ and $\boxplus$ are the only metric operators allowed in rule heads. Thus, if r is a rule in Π_s^σ whose body holds in some interpretation at a time point t, then there is a unique way of extending this interpretation with facts over the predicate P' in the head of r so that the head holds at t. Since $\sigma(P') = s$, the new interpretation agrees with the original interpretation on predicates P with $\sigma(P) \neq s$. The other two conditions hold by construction. $\square$

It is worth noting that the second condition of Proposition 4 does not imply the first one, since $\mathfrak{I}$ may interpret predicates which are defined by the s-th stratum. Let $\mathfrak{L}_s^\sigma(\Pi, \mathfrak{I})$ denote the unique least interpretation guaranteed by Proposition 4, which we will use to define a sequence of partial materialisations. Their definition is similar to Datalog; however in DatalogMTL$^\neg$ each partial materialisation is potentially infinite.

Definition 5. *Let Π be a program with a stratification σ and let $\mathcal{D}$ be a dataset. Then, for each $s \in \mathbb{N}$, the s-th* partial materialisation *$\mathfrak{M}_s^\sigma(\Pi, \mathcal{D})$ of Π and $\mathcal{D}$ with respect to σ is*

- *the least model of $\mathcal{D}$, if $s = 0$, and*
- *the interpretation $\mathfrak{L}_s^\sigma(\Pi, \mathfrak{M}_{s-1}^\sigma(\Pi, \mathcal{D}))$, if $s > 0$.*

The materialisation *$\mathfrak{M}^\sigma(\Pi, \mathcal{D})$ of Π and $\mathcal{D}$ with respect to σ is $\mathfrak{M}_n^\sigma(\Pi, \mathcal{D})$, for n the greatest number in the range of σ (or 0, when $\Pi = \emptyset$ and hence the range of σ is empty).*

Clearly, the materialisation $\mathfrak{M}^\sigma(\Pi, \mathcal{D})$ is a model of $\mathcal{D}$ as $\mathfrak{M}_0^\sigma(\Pi, \mathcal{D})$ already is. Moreover, $\mathfrak{M}^\sigma(\Pi, \mathcal{D})$ is a model of Π as, by the construction, each $\mathfrak{M}_s^\sigma(\Pi, \mathcal{D})$ is a model of $\Pi_{\leq s}^\sigma$.

The following proposition says that, as in Datalog (Abiteboul, Hull, and Vianu 1995), our materialisations are independent of the choice of a stratification.

Proposition 6. *For each program Π, dataset $\mathcal{D}$, and stratifications σ_1 and σ_2 of Π, we have $\mathfrak{M}^{\sigma_1}(\Pi, \mathcal{D}) = \mathfrak{M}^{\sigma_2}(\Pi, \mathcal{D})$.*

Thus, in what follows, given a stratifiable program Π we will restrict attention to its single stratification σ_Π that assigns the smallest possible number to each predicate (similarly to Datalog, such a stratification is uniquely defined). Furthermore, we will say that Π *admits* n strata if n is the greatest number in the range of σ_Π (or 0, when $\Pi = \emptyset$).

In the remainder of the paper we assume that σ_Π is implicit in all our definitions and technical results, and we will no longer mention the stratifications explicitly; for example, we write $\mathfrak{M}(\Pi, \mathcal{D})$ instead of $\mathfrak{M}^{\sigma_\Pi}(\Pi, \mathcal{D})$.

We are ready to define fact entailment in DatalogMTL$^\neg$.

Definition 7. *A stratifiable program* Π *and a dataset* $\mathcal{D}$ entail *a metric fact* $M@\varrho$*, written* $(\Pi, \mathcal{D}) \models M@\varrho$*, if* $\mathfrak{M}(\Pi, \mathcal{D}) \models M@\varrho$.

We conclude this section by showing that our semantics of stratifiable programs extends the semantics of positive programs by Brandt et al. (2018).

Proposition 8. *Let* Π *be a positive program,* $\mathcal{D}$ *a dataset, and* $M@\varrho$ *a metric fact. Then,* $\mathfrak{M}(\Pi, \mathcal{D}) \models M@\varrho$ *if and only if every model of* Π *and* $\mathcal{D}$ *is also a model of* $M@\varrho$.

Proof. It is known that each pair consisting of a positive program and a dataset has a unique least model, referred to as the *canonical interpretation* (Brandt et al. 2018). Hence, the canonical interpretation is a model of $M@\varrho$ if and only if so is each model of Π and $\mathcal{D}$. Since Π is positive it admits exactly one stratum, so $\mathfrak{M}(\Pi, \mathcal{D}) = \mathfrak{M}_1(\Pi, \mathcal{D})$. By Definition 5, $\mathfrak{M}_1(\Pi, \mathcal{D})$ is the least interpretation satisfying $\mathcal{D}$ and Π; so it coincides with the canonical interpretation of Π and $\mathcal{D}$, which implies the claim. $\square$

Automata for Partial Materialisations

We next introduce generalised Büchi automata that will allow us to check which facts hold in a partial materialisation.

In this section we restrict ourselves to programs in a normal form where each rule satisfies the following additional requirements: its head is a relational atom, there is neither nesting of metric operators nor occurrences of $\diamondminus$ and $\diamondplus$ in its body, and the left endpoints of all unbounded intervals it mentions are 0. This normal form extends that proposed by Wałęga et al. (2019) for positive programs.

Proposition 9. *Each stratifiable program* Π *can be transformed in polynomial time into a stratifiable program* Π' *in normal form such that, for each dataset* $\mathcal{D}$ *and relational fact* $P(\mathbf{c})@\varrho$ *with* P *in the signature of* Π*, we have* $(\Pi, \mathcal{D}) \models P(\mathbf{c})@\varrho$ *if and only if* $(\Pi', \mathcal{D}) \models P(\mathbf{c})@\varrho$.

Thus, from now, we assume that each program is in normal form. Moreover, until the end of the section we assume that Π is a stratifiable program admitting n strata and $\mathcal{D}$ is a dataset. We will use the following notions.

Definition 10. *Let* $\mathsf{div}(\Pi) = \frac{1}{k}$*, where* k *is the product of all denominators in rational endpoints of intervals in* Π *(for definiteness we set* $k = 1$ *if* Π *has no intervals with rational endpoints). The* $(\Pi, \mathcal{D})$-ruler *is the set of all time points of the form* $t + i \cdot \mathsf{div}(\Pi)$*, for* t *a rational number mentioned in* $\mathcal{D}$ *and* i *an integer. A* $(\Pi, \mathcal{D})$-interval *is either a (punctual) interval containing a single point in the* $(\Pi, \mathcal{D})$*-ruler, or an interval* (t_1, t_2)*, where* t_1 *and* t_2 *are consecutive points in the* $(\Pi, \mathcal{D})$*-ruler. An interpretation* $\mathfrak{I}$ *is a* $(\Pi, \mathcal{D})$-interpretation *if it satisfies the following property for each* $t \in \mathbb{Q}$ *and for each ground relational atom* $P(\mathbf{c})$*: if* $\mathfrak{I}, t \models P(\mathbf{c})$*, then* $\mathfrak{I}, t' \models P(\mathbf{c})$ *for each* t' *in the same* $(\Pi, \mathcal{D})$*-interval as* t.

To ensure that the $(\Pi, \mathcal{D})$-ruler and the set of $(\Pi, \mathcal{D})$-intervals are non-empty, we assume that each dataset mentions at least one rational number (which can always be achieved by introducing a dummy relational fact in $\mathcal{D}$). It is known that if Π is positive then the unique least common model of Π and $\mathcal{D}$ (called their materialisation) is a $(\Pi, \mathcal{D})$-interpretation (Wałęga et al. 2019). We generalise this result to DatalogMTL$^\neg$ programs and all partial materialisations.

Lemma 11. *For every* $s \in \{0, \dots, n\}$*, the* s*-th partial materialisation is a* $(\Pi, \mathcal{D})$*-interpretation.*

Proof sketch. We proceed by induction on s. The base case holds since $\mathfrak{M}_0(\Pi, \mathcal{D})$ is the least model of $\mathcal{D}$ and all rationals mentioned in facts of $\mathcal{D}$ are in the $(\Pi, \mathcal{D})$-ruler. The inductive step holds by a similar argument to the one used by Wałęga et al. (2019) to show that the canonical interpretation of a positive program Π' and a dataset $\mathcal{D}'$ is a $(\Pi', \mathcal{D}')$-interpretation. $\square$

In the remainder of this section, for every $s \in \{1, \dots, n\}$, we will define a family of pairs of generalised Büchi automata that will allow us to check which metric facts hold in the s-th partial materialisation. States of these automata, called *windows*, describe which metric facts are satisfied in specified bounded intervals. Then, for each initial window $\mathcal{W}_0$, we will introduce two automata whose accepting runs determine, by means of metric facts in their states, all $(\Pi, \mathcal{D})$-interpretations that simultaneously satisfy all facts in $\mathcal{W}_0$, extend $\mathfrak{M}_{s-1}(\Pi, \mathcal{D})$, agree with $\mathfrak{M}_{s-1}(\Pi, \mathcal{D})$ on predicates not defined by Π_s, and are models of Π_s. In particular, each such interpretation can be divided into a fragment to the left (i.e., in the past) of $\mathcal{W}_0$, the window $\mathcal{W}_0$ itself, and a fragment to the right (i.e., in the future) of $\mathcal{W}_0$. The first automaton is responsible for determining the left fragments of the interpretations, while the second automaton for the right fragments. Then, $\mathfrak{M}_s(\Pi, \mathcal{D})$ is the least among interpretations determined by these automata, and thus, we can check if a fact holds in $\mathfrak{M}_s(\Pi, \mathcal{D})$ by examining accepting runs of all relevant pairs of our automata.

We start by defining ground metric atoms which will occur in windows, and so, which will be used to reconstruct interpretations from accepting runs.

Definition 12. *Given* $\Pi' \subseteq \Pi$*,* $\mathsf{gr}(\Pi', \mathcal{D})$ *is the set of all ground rules that can be obtained from rules in* Π' *by assigning variables to constants from* Π' *and* $\mathcal{D}$*. Then,* $\mathsf{gma}(\Pi', \mathcal{D})$ *is the set of all relational atoms in* $\mathcal{D}$*, all metric atoms mentioned in rules in* $\mathsf{gr}(\Pi', \mathcal{D})$*, and all metric atoms of the form* $\boxminus_{\langle 0, \infty)} P(\mathbf{c})$ *and* $\boxplus_{\langle 0, \infty)} P(\mathbf{c})$*, with* $P(\mathbf{c})$ *a relational atom occurring in the rules in* $\mathsf{gr}(\Pi', \mathcal{D})$.

We are ready to define the windows.

Definition 13. *For* $s \in \{1, \dots, n\}$*, a* $(\Pi, \mathcal{D})_s$-window *is a pair* (ϱ, W)*, where* ϱ *is a non-empty interval with all its rational endpoints in the* $(\Pi, \mathcal{D})$*-ruler, and* W *is a set of metric facts* $M@\varrho'$*, with* $M \in \mathsf{gma}(\Pi_{\leq s}, \mathcal{D})$ *and* $\varrho' \subseteq \varrho$ *a* $(\Pi, \mathcal{D})$*-interval, satisfying the following conditions:*

– *for every* $M \in \mathsf{gma}(\Pi_{\leq s-1}, \mathcal{D})$ *and every* $(\Pi, \mathcal{D})$*-interval* $\varrho' \subseteq \varrho$*, we have* $M@\varrho' \in W$ *if and only if* $\mathfrak{M}_{s-1}(\Pi, \mathcal{D}) \models M@\varrho'$*,*

– *there is a $(\Pi, \mathcal{D})$-interpretation $\mathfrak{I}$ such that, for every $M \in \mathsf{gma}(\Pi_{\leq s}, \mathcal{D})$ and every $(\Pi, \mathcal{D})$-interval $\varrho' \subseteq \varrho$, we have $M@\varrho' \in W$ if and only if $\mathfrak{I} \models M@\varrho'$.*

The length *of a window is the (possibly infinite) number of $(\Pi, \mathcal{D})$-intervals contained in ϱ.*

The first item in the definition ensures that the restriction of a $(\Pi, \mathcal{D})_s$-window (ϱ, W) to predicates not defined by Π_s agrees with $\mathfrak{M}_{s-1}(\Pi, \mathcal{D})$. The second item ensures that W can be extended to an interpretation (which is not necessarily $\mathfrak{M}_s(\Pi, \mathcal{D})$).

The following definition describes the class of windows that will constitute the states of our automata, where we say that a window is *over* an interval ϱ if it is of the form (ϱ, W).

Definition 14. *A $(\Pi, \mathcal{D})_s$-window (ϱ, W)* locally satisfies *Π_s if, for each ground rule of Form (1) in $\mathsf{gr}(\Pi_s, \mathcal{D})$, and each $(\Pi, \mathcal{D})$-interval $\varrho' \subseteq \varrho$, we have $M@\varrho' \in W$ whenever $M_i@\varrho' \in W$ for each $i \in \{1, \ldots, k\}$ and $M_i@\varrho' \notin W$ for each $i \in \{k+1, \ldots, k+m\}$.*

We are now ready to define the automata. To this end, let the *left-shift* $\overleftarrow{\varrho}$ of a bounded interval ϱ with both endpoints in the $(\Pi, \mathcal{D})$-ruler be the interval $(\varrho' \cup \varrho) \setminus \varrho''$ with ϱ' the first $(\Pi, \mathcal{D})$-interval to the left of ϱ and ϱ'' the right-most $(\Pi, \mathcal{D})$-interval contained in ϱ; then let the *right-shift* $\overrightarrow{\varrho}$ of ϱ be defined symmetrically.

Definition 15. *Let $s \in \{1, \ldots, n\}$ and let $\mathcal{W}_0 = (\varrho_0, W_0)$ be a $(\Pi, \mathcal{D})_s$-window of finite length locally satisfying Π_s. The* left automaton $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ for Π_s *is the generalised Büchi automaton $(\mathcal{Q}, \Sigma, \delta, \mathcal{W}_0, \mathcal{F})$ with the following components:*

– *the set $\mathcal{Q}$ of states is the set of all $(\Pi, \mathcal{D})_s$-windows of the same length as $\mathcal{W}_0$ locally satisfying Π_s;*
– *the alphabet Σ is the powerset of $\mathsf{gma}(\Pi_{\leq s}, \mathcal{D})$;*
– *the transition function δ is a partial function from $\mathcal{Q} \times \Sigma$ to $\mathcal{Q}$ such that, for each state $\mathcal{W} = (\varrho, W)$ and each $A \in \Sigma$, the value $\delta(\mathcal{W}, A)$ is the pair*

$$(\overleftarrow{\varrho},\ \{M@\varrho' \in W \mid \varrho' \subseteq \varrho \cap \overleftarrow{\varrho}\} \cup \{M@(\overleftarrow{\varrho} \setminus \varrho) \mid M \in A\}),$$

if this pair is a state, and undefined otherwise;
– *$\mathcal{W}_0$ is the initial state;*
– *the accepting condition $\mathcal{F}$ is a family containing, for each $\boxminus_{\varrho_\infty} M \in \mathsf{gma}(\Pi_{\leq s}, \mathcal{D})$ with $\varrho_\infty = \langle 0, \infty)$, the set*

$$\{(\varrho, W) \in \mathcal{Q} \mid \text{there is a } (\Pi, \mathcal{D})\text{-interval } \varrho' \subseteq \varrho \text{ such that } \boxminus_{\varrho_\infty} M@\varrho' \in W \text{ or } M@\varrho' \notin W\},$$

and, for each $M_1 \mathcal{S}_{\varrho_\infty} M_2 \in \mathsf{gma}(\Pi_{\leq s}, \mathcal{D})$, the set

$$\{(\varrho, W) \in \mathcal{Q} \mid \text{there is a } (\Pi, \mathcal{D})\text{-interval } \varrho' \subseteq \varrho \text{ such that } M_1 \mathcal{S}_{\varrho_\infty} M_2@\varrho' \notin W \text{ or } M_2@\varrho' \in W\}.$$

The automaton $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ accepts an infinite Σ-word $A_0 A_1 \ldots$ if there is a sequence $\mathcal{W}_0, \mathcal{W}_1, \ldots$ of states, called an accepting run*, such that $\mathcal{W}_{i+1} = \delta(\mathcal{W}_i, A_i)$ for each $i \in \mathbb{N}$, and the sequence contains, for each $F \in \mathcal{F}$, an infinite number of occurrences of states belonging to F.*

The right automaton $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ *for Π_s is defined in the same way, except that in the definition of δ the right-shift $\overrightarrow{\varrho}$ is used instead of the left-shift $\overleftarrow{\varrho}$, and in the definition of $\mathcal{F}$, $\boxplus$ and $\mathcal{U}$ are used instead of $\boxminus$ and $\mathcal{S}$, respectively.*

As we will show, for a $(\Pi, \mathcal{D})_s$-window $\mathcal{W}_0 = (\varrho_0, W_0)$, the pairs of accepting runs of $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ for Π_s determine all $(\Pi, \mathcal{D})$-interpretations which are models of Π_s as well as extend $\mathfrak{M}_{s-1}(\Pi, \mathcal{D})$ and agree with it on predicates not defined by Π_s. By Definition 5, each such model contains $\mathfrak{M}_s(\Pi, \mathcal{D})$, so a fact holds in $\mathfrak{M}_s(\Pi, \mathcal{D})$ if it holds in all interpretations determined by $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$.

To guarantee that the pairs of accepting runs represent interpretations satisfying Π_s, we need to ensure that the initial window $\mathcal{W}_0$ is not too short. Otherwise, even when each state in an accepting run locally satisfies Π_s, the interpretation represented by the whole run may not satisfy Π_s. In particular the length of $\mathcal{W}_0$ cannot be smaller than the number of $(\Pi, \mathcal{D})$-intervals contained in $\varrho_{(\Pi,\mathcal{D})} = [t_\mathcal{D}, t_\mathcal{D} + t_\Pi]$, where t_Π and $t_\mathcal{D}$ are the largest rational numbers mentioned in Π and $\mathcal{D}$, respectively (if Π has no intervals with rational endpoints, then we take $t_\Pi = 1$ for definiteness).

Theorem 16. *The following are equivalent for each $s \in \{1, \ldots, n\}$, $M \in \mathsf{gma}(\Pi_{\leq s}, \mathcal{D})$, and $(\Pi, \mathcal{D})$-interval ϱ:*

1. *$\mathfrak{M}_s(\Pi, \mathcal{D}) \models M@\varrho$;*
2. *for each $(\Pi, \mathcal{D})_s$-window $\mathcal{W}_0$ over $\varrho_{(\Pi,\mathcal{D})}$ locally satisfying Π_s and each pair of accepting runs of the automata $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ for Π_s, there is a state (ϱ', W') in the runs such that $M@\varrho \in W'$.*

Proof sketch. Assume $\mathcal{W}_0 = (\varrho_{(\Pi,\mathcal{D})}, W_0)$ is a $(\Pi, \mathcal{D})_s$-window. First, we will show that if $\mathcal{W}_0, \mathcal{W}_{-1}, \ldots$ and $\mathcal{W}_0, \mathcal{W}_1, \ldots$, with $\mathcal{W}_i = (\varrho_i, W_i)$, are accepting runs of the automata $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$, respectively, then their 'union' $\mathcal{W}_\infty = ((-\infty, \infty), \bigcup_{i \in \mathbb{Z}} W_i)$ is a $(\Pi, \mathcal{D})_s$-window (of infinite length) locally satisfying Π_s. The union $\mathcal{W}_\infty$ satisfies the first item from Definition 13 trivially since each $\mathcal{W}_i$ already satisfies it. To show that it also satisfies the second item, we first define the *interpretation $\mathfrak{I}$ of $\mathcal{W}_\infty$* as follows: for each relational atom $P(\mathbf{c})$ and $(\Pi, \mathcal{D})$-interval ϱ we have $\mathfrak{I} \models P(\mathbf{c})@\varrho$ if and only if $P(\mathbf{c})@\varrho \in \bigcup_{i \in \mathbb{Z}} W_i$. We can show that $\mathfrak{I}$ is a witness for the second item from Definition 13. Indeed, we can show that the condition therein holds for metric atoms with bounded intervals using the fact that Π is in normal form and each window in an accepting run is long enough (since it has the same length as $\varrho_{(\Pi,\mathcal{D})}$). For atoms with unbounded intervals, we use the fact that the accepting runs satisfy the accepting conditions of the automata, which are designed to mimic the semantics of operators with unbounded intervals. Once this has been established, we can easily prove that $\mathcal{W}_\infty$ locally satisfies Π_s using the fact that so does each window in the runs.

Let $X_{\mathcal{W}_0}$ be the set containing each $(\Pi, \mathcal{D})$-interpretation that satisfies the following conditions: (i) it is a model of W_0, (ii) it contains $\mathfrak{M}_{s-1}(\Pi, \mathcal{D})$, (iii) it agrees with $\mathfrak{M}_{s-1}(\Pi, \mathcal{D})$ on all predicates P with $\sigma_\Pi(P) \neq s$, and (iv) it is a model of Π_s. We claim that ($\star$) an interpretation $\mathfrak{I}$ belongs to $X_{\mathcal{W}_0}$ if and only if there are accepting runs $\mathcal{W}_0, \mathcal{W}_{-1}, \ldots$ and $\mathcal{W}_0, \mathcal{W}_1, \ldots$ of $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$, respectively, such that

$\mathfrak{I}$ is the interpretation of $\mathcal{W}_\infty = ((-\infty, \infty), \bigcup_{i \in \mathbb{Z}} W_i)$, for $\mathcal{W}_i = (\varrho_i, W_i)$. Indeed, if $\mathfrak{I} \in X_{\mathcal{W}_0}$, then it is straightforward to show that there exists a unique $(\Pi, \mathcal{D})_s$-window of infinite length locally satisfying Π_s such that $\mathfrak{I}$ is the interpretation of $\mathcal{W}_\infty$. Then, we can divide $\mathcal{W}_\infty$ into windows of the same length as $\mathcal{W}_0$ to obtain states in accepting runs of $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$. For the opposite direction, let $\mathfrak{I}$ be the interpretation of $\mathcal{W}_\infty$. We need to show that $\mathfrak{I}$ satisfies Conditions (i)–(iv). Clearly, $W_0 \subseteq \bigcup_{i \in \mathbb{Z}} W_i$, so Condition (i) holds. Conditions (ii) and (iii) hold directly by the fact that each $\mathcal{W}_i$ is a $(\Pi, \mathcal{D})_s$-window. As we have shown, $\mathcal{W}_\infty$ is a $(\Pi, \mathcal{D})_s$-window locally satisfying Π_s, so $\mathfrak{I}$ is a model of Π_s. Thus Condition (iv) also holds.

By Proposition 4 and Definition 5, $\mathfrak{M}_s(\Pi, \mathcal{D})$ is the unique least interpretation satisfying Conditions (ii)–(iv). Moreover, by Lemma 11, $\mathfrak{M}_s(\Pi, \mathcal{D})$ is a $(\Pi, \mathcal{D})$-interpretation, and so $\mathfrak{M}_s(\Pi, \mathcal{D}) \models M@\varrho$ if and only if $\mathfrak{I} \models M@\varrho$ for each $(\Pi, \mathcal{D})_s$-window $\mathcal{W}_0 = (\varrho_{(\Pi,\mathcal{D})}, W_0)$ and $\mathfrak{I} \in X_{\mathcal{W}_0}$. The latter is equivalent to the second statement in the theorem by Claim $(\star)$. □

Theorem 16 suggests the following method of checking whether $\mathfrak{M}_s(\Pi, \mathcal{D}) \models M@\varrho$. Consider the (finite) set X of all $(\Pi, \mathcal{D})_s$-windows $\mathcal{W}_0$ over $\varrho_{(\Pi,\mathcal{D})}$. Then, check if there is $\mathcal{W}_0 \in X$ such that $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ for Π_s have accepting runs but $M@\varrho$ does not occur in these runs. If so, then $\mathfrak{M}_s(\Pi, \mathcal{D}) \not\models M@\varrho$; otherwise, $\mathfrak{M}_s(\Pi, \mathcal{D}) \models M@\varrho$. By Definitions 5 and 7, checking whether Π and $\mathcal{D}$ entail $M@\varrho$ reduces to checking if $\mathfrak{M}_n(\Pi, \mathcal{D}) \models M@\varrho$ which, as we have argued, is feasible using our automata.

There are, however, two main obstacles to use this approach. First, each automaton has infinitely many states (since there are infinitely many intervals containing the same number of $(\Pi, \mathcal{D})$-intervals as $\varrho_{(\Pi,\mathcal{D})}$); as we will show, this problem can be addressed by defining an equivalence relation on the states with a finite quotient set. A more significant problem is how to determine states of the automata for Π_s. Such states need to agree with the previous partial materialisation, which we cannot explicitly construct as it may be infinite. We will show in the next section that this problem can be overcome by constructing on the fly only essential fragments of partial materialisations. This will give us a worst-case optimal procedure for checking fact entailment.

Computational Complexity

Now, we will use the automata from the previous section to establish the computational complexity of *fact entailment* in DatalogMTL$^\neg$, namely, the problem of checking whether a relational fact is entailed by a stratifiable program and a dataset. We will consider the *combined complexity* of the problem, where the fact, the program, and the dataset all form an input, and two variants of the *data complexity*, which is a standard measure in data-intensive applications: in the first variant, only the dataset is treated as an input while the program and the fact are fixed; in the second, the fact is also a part of an input. We will first show that both variants of the data complexity of fact entailment are in PSPACE. Then, we will show that the combined complexity of the problem is in EXPSPACE. The matching lower bounds hold already for positive programs (Wałęga et al. 2019; Brandt et al. 2017) and so our bounds are tight.

First, we will show that there exists a polynomial space procedure that uses the automata from the previous section to construct a fragment of a partial materialisation restricted to a given (bounded) interval. Such fragments can be seen as windows of a special form, as defined below.

Definition 17. *Let* Π *be a stratifiable program which admits* n *strata, let* $s \in \{1, \dots, n\}$, *and let* $\mathcal{D}$ *be a dataset. A* materialisation $(\Pi, \mathcal{D})_s$-window *is a* $(\Pi, \mathcal{D})_s$*-window that satisfies a modification of the first item in Definition 13, where occurrences of* $s - 1$ *are replaced with* s.

It follows that for every interval ϱ whose rational endpoints are in the $(\Pi, \mathcal{D})$-ruler, there exists a unique materialisation $(\Pi, \mathcal{D})_s$-window over ϱ.

Lemma 18. *Let* Π *be a stratifiable program admitting* n *strata, let* $s \in \{1, \dots, n\}$, *let* $\mathcal{D}$ *be a dataset, and let* ϱ *be a bounded interval whose endpoints are in the* $(\Pi, \mathcal{D})$*-ruler. Computing the materialisation* $(\Pi, \mathcal{D})_s$*-window over* ϱ *is feasible in exponential space in general and in polynomial space in the size of (the representations of)* $\mathcal{D}$ *and* ϱ.

Proof sketch. We address the polynomial space case, while the argument for the general case is similar. We show, by mutual induction on $s \in \{1, \dots, n\}$, the following two claims: (i) given Π, n, s, and $\mathcal{D}$ as in the lemma, and a $(\Pi, \mathcal{D})_s$-window $\mathcal{W}_0$ of finite length locally satisfying Π_s, checking whether the languages of $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ and $\mathcal{A}^{\rightarrow}_{\mathcal{W}_0}$ for Π_s are non-empty is feasible in polynomial space in the size of $\mathcal{D}$ and $\mathcal{W}_0$, and (ii) the statement from the lemma.

To show the base case for Claim (i), let $\mathcal{W}_0 = (\varrho_0, W_0)$ be a $(\Pi, \mathcal{D})_1$-window of a finite length locally satisfying Π_1. Let $\varrho_0, \varrho_1, \dots$ be a sequence of intervals such that $\varrho_{i+1} = \overleftarrow{\varrho_i}$ and let ϱ_k be the first interval in this sequence which is to the left of all rational numbers mentioned in ϱ_0 and $\mathcal{D}$. Clearly, the left automaton $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ for Π_1 has an accepting run if and only if there is a state $\mathcal{W}_k = (\varrho_k, W_k)$ such that $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ has a run from $\mathcal{W}_0$ to $\mathcal{W}_k$, and the left automaton $\mathcal{A}^{\leftarrow}_{\mathcal{W}_k}$ for Π_1 has an accepting run. We show how to check these two conditions in PSPACE.

First, we guess $\mathcal{W}_k$, which is feasible in PSPACE due to the fact that the endpoints of ϱ_k have polynomial representations. By definition, $\mathcal{W}_k$ is a state of $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ if it is a $(\Pi, \mathcal{D})_1$-window locally satisfying Π_1, which can be checked in PSPACE by first scanning $\mathcal{D}$ to verify the first item in Definition 13, then guessing a polynomial representation of an interpretation satisfying the second item of Definition 13, and finally verifying that $\mathcal{W}_k$ locally satisfies Π_1. Next, we guess states in a run of $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ from $\mathcal{W}_0$ to $\mathcal{W}_k$ one by one. As in the case of $\mathcal{W}_k$, each of them can be guessed and checked against being a state in PSPACE. Furthermore, checking if $\mathcal{A}^{\leftarrow}_{\mathcal{W}_0}$ has transitions between consecutive states is also feasible in PSPACE. It remains to check if $\mathcal{A}^{\leftarrow}_{\mathcal{W}_k}$ has an accepting run. This cannot be done by guessing states, since endpoints of intervals ϱ_i with $i > k$ can have arbitrarily long representations. Instead, we use the idea of Wałęga et al. (2019) to construct a finite Büchi automaton equivalent to $\mathcal{A}^{\leftarrow}_{\mathcal{W}_k}$ (by merging states of $\mathcal{A}^{\leftarrow}_{\mathcal{W}_k}$ which differ only by a shift of intervals they mention) which has exponentially many states, each polynomial in size, and for which checking non-emptiness of its language is feasible in PSPACE. The non-emptiness of the language of the right automaton $\mathcal{A}_{\mathcal{W}_0}^{\rightarrow}$ for Π_1 can be checked in PSPACE analogously.

Now, we show the base case for Claim (ii) by constructing in PSPACE the materialisation $(\Pi, \mathcal{D})_1$-window over ϱ. Therefore, for every $M@\varrho'$ such that $M \in \mathsf{gma}(\Pi_{\leq 1}, \mathcal{D})$ and $\varrho' \subseteq \varrho$ is a $(\Pi, \mathcal{D})$-interval, we need to check whether $\mathfrak{M}_1(\Pi, \mathcal{D}) \models M@\varrho'$. There are exponentially many such $M@\varrho'$ and for each of them we perform the check separately as follows. Let ϱ'' be the (unique) interval with endpoints in the $(\Pi, \mathcal{D})$-ruler that contains the same number of $(\Pi, \mathcal{D})$-intervals as $\varrho_{(\Pi,\mathcal{D})}$ and such that ϱ' is the leftmost $(\Pi, \mathcal{D})$-interval contained in ϱ''. Then, we can use Theorem 16 to show that $\mathfrak{M}_1(\Pi, \mathcal{D}) \not\models M@\varrho'$ if and only if there is a $(\Pi, \mathcal{D})_1$-window $\mathcal{W} = (\varrho'', W)$ locally satisfying Π_1 such that the languages of $\mathcal{A}_{\mathcal{W}}^{\leftarrow}$ and $\mathcal{A}_{\mathcal{W}}^{\rightarrow}$ for Π_1 are non-empty and $M@\varrho' \notin W$, which as we have shown in the base case for Claim (i), can be checked in PSPACE in the size of $\mathcal{D}$ and $\mathcal{W}$. The interval ϱ'' is polynomially representable in the size of $\mathcal{D}$ and ϱ, and so is $\mathcal{W}$, thus construction of the materialisation $(\Pi, \mathcal{D})_1$-window over ϱ is in PSPACE in the size of $\mathcal{D}$ and ϱ.

In the inductive step we show that Claims (i) and (ii) hold for $s > 1$ if they hold for $s - 1$. To show Claim (i) let $\mathcal{W}_0 = (\varrho_0, W_0)$ be a $(\Pi, \mathcal{D})_s$-window of a finite length locally satisfying Π_s. To check if $\mathcal{A}_{\mathcal{W}_0}^{\leftarrow}$ and $\mathcal{A}_{\mathcal{W}_0}^{\rightarrow}$ for Π_s have accepting runs we proceed as in the base case. Though, we face an additional obstacle while checking whether a guessed $\mathcal{W}_i = (\varrho_i, W_i)$ is a $(\Pi, \mathcal{D})_s$-window, as verifying the first item of Definition 13 no longer can be achieved by scanning $\mathcal{D}$. Instead, we need to construct the materialisation $(\Pi, \mathcal{D})_{s-1}$-window over ϱ_i, which by the inductive hypothesis for Claim (ii), is feasible in PSPACE. In a similar way we can overcome the problem of checking in PSPACE whether the finite automaton obtained from $\mathcal{A}_{\mathcal{W}_k}^{\leftarrow}$ over Π_s has an accepting run. Verifying that $\mathcal{A}_{\mathcal{W}_0}^{\rightarrow}$ has an accepting run is symmetric.

Finally, we show the inductive step for Claim (ii) by constructing the materialisation $(\Pi, \mathcal{D})_s$-window over ϱ. First, we construct the materialisation $(\Pi, \mathcal{D})_{s-1}$ window over ϱ, which is feasible in PSPACE by the inductive hypothesis for Claim (ii). Then we check, for each $M@\varrho'$ such that $M \in \mathsf{gma}(\Pi_s, \mathcal{D})$ and $\varrho' \subseteq \varrho$ is a $(\Pi, \mathcal{D})$-interval, whether $\mathfrak{M}_s(\Pi, \mathcal{D}) \models M@\varrho'$. We can do this using Theorem 16 in a similar way as in the base case, except that now we use automata for Π_s and not for Π_1. By the inductive step for Claim (i) checking if such automata have non-empty languages is in PSPACE, and so the whole procedure can be performed in polynomial space. $\square$

Next, we will use Lemma 18 to establish tight bounds for (both variants of) the data complexity of fact entailment.

Theorem 19. *Checking fact entailment in* DatalogMTL$^\neg$ *is* PSPACE*-complete in both variants of the data complexity.*

Proof. The lower bound follows from Proposition 8 and from the PSPACE-completeness in the data complexity of fact entailment for positive programs (Wałęga et al. 2019).

For the upper bound, it is sufficient to concentrate on the variant of data complexity where a fact is part of the input. Assume that we are checking if a relational fact $P(\mathbf{c})@\varrho$ is entailed by a stratifiable program Π admitting n strata and a dataset $\mathcal{D}$. For simplicity, we assume that $P(\mathbf{c}) \in \mathsf{gr}(\Pi, \mathcal{D})$ (which does not increase complexity) and that ϱ is bounded (otherwise we need to use metric atoms $\boxminus_{\langle 0,\infty)} P(\mathbf{c})$ and $\boxplus_{\langle 0,\infty)} P(\mathbf{c})$, which are guaranteed to occur in $\mathsf{gma}(\Pi, \mathcal{D})$). Let X be the least set of $(\Pi, \mathcal{D})$-intervals whose union entirely covers ϱ. By Lemma 11, checking if $(\Pi, \mathcal{D}) \models P(\mathbf{c})@\varrho$ reduces to verifying if $(\Pi, \mathcal{D}) \models P(\mathbf{c})@\varrho'$ for each $(\Pi, \mathcal{D})$-interval $\varrho' \in X$. Since ϱ is bounded, X is finite and each ϱ' is polynomially representable. Now, for $\varrho' \in X$, let ϱ'' be the interval with endpoints in the $(\Pi, \mathcal{D})$-ruler that contains the same number of $(\Pi, \mathcal{D})$-intervals as $\varrho_{(\Pi,\mathcal{D})}$ and such that ϱ' is the leftmost $(\Pi, \mathcal{D})$-interval contained in ϱ''. It remains to construct the materialisation $(\Pi, \mathcal{D})_n$-window (ϱ'', W) and check if $P(\mathbf{c})@\varrho' \in W$, which by Lemma 18 is feasible in polynomial space. $\square$

Now, we will consider the combined complexity; in this case the program is a part of the input, which results in an increase of the complexity.

Theorem 20. *Checking fact entailment in* DatalogMTL$^\neg$ *is* EXPSPACE*-complete in combined complexity.*

Proof. The lower bound holds already for positive programs (Brandt et al. 2017). To prove the upper bound, we use the same procedure as in the proof of Theorem 19. The main difference is that, since a program is a part of an input, the materialisation $(\Pi, \mathcal{D})_n$-window (ϱ'', W) constructed therein is now exponentially big. Indeed, the number of $(\Pi, \mathcal{D})$-intervals contained in ϱ'' (as well as in $\varrho_{(\Pi,\mathcal{D})}$) is exponential and the number of facts over each $(\Pi, \mathcal{D})$-interval contained in ϱ'' also can be exponential. This yields the EXPSPACE upper bound. $\square$

Conclusions and Future Work

We have introduced DatalogMTL$^\neg$, which is a first extension of DatalogMTL with non-monotonic (in particular stratified) negation. The obtained language allows us to perform reasoning over the rational timeline, with use of metric operators, and negative information, which makes it attractive for many practical applications. As we have shown, such an extension of DatalogMTL does not increase its complexity: reasoning remains PSPACE- and EXPSPACE-complete in data and combined complexity, respectively.

It is worth noting that our language can be treated as an extension of Datalog$_{1S}$ with stratified negation, which allows metric operators (instead of only the successor operator) and is interpreted over the rational (and not over the integer) timeline. Thus, our complexity results imply the same upper bounds for Datalog$_{1S}$ with stratified negation.

In future we plan to investigate syntactical fragments of DatalogMTL$^\neg$, for example similar to those constructed by Wałęga et al. (2020). We also want to consider other semantics for non-monotonic negation by exploiting local and temporal stratifications, and stable models.

Acknowledgements

This work was supported by the AIDA project (Alan Turing Institute), the SIRIUS Centre for Scalable Data Access (Research Council of Norway), and the EPSRC projects OASIS (EP/S032347/1), AnaLOG (EP/P025943/1), and UK FIRES (EP/S019111/1).

References

Abadi, M.; and Manna, Z. 1989. Temporal Logic Programming. *Journal of Symbolic Computation* 8(3): 277–295.

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Reading, Mass.: Addison-Wesley.

Beck, H.; Dao-Tran, M.; and Eiter, T. 2018. LARS: A Logic-Based Framework for Analytic Reasoning over Streams. *Artificial Intelligence* 261: 16–70.

Brandt, S.; Kalaycı, E. G.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2018. Querying Log Data with Metric Temporal Logic. *Journal of Artificial Intelligence Research* 62: 829–877.

Brandt, S.; Kontchakov, R.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2017. Ontology-Based Data Access with a Horn Fragment of Metric Temporal Logic. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 1070–1076. AAAI Press.

Brzoska, C. 1998. Programming in Metric Temporal Logic. *Theoretical Computer Science* 202(1–2): 55–125.

Cabalar, P.; Dieguez, M.; Schaub, T.; and Schuhmann, A. 2020. Towards Metric Temporal Answer Set Programming. arXiv preprint. arXiv:2008.02038 [cs.AI].

Cabalar, P.; Kaminski, R.; Morkisch, P.; and Schaub, T. 2019. Telingo = ASP + Time. In *Proceedings of the 15th International Conference on Logic Programming and Nonmonotonic Reasoning*, 256–269. Springer.

Chomicki, J.; and Imieliński, T. 1988. Temporal Deductive Databases and Infinite Objects. In *Proceedings of the 7th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 61–73. ACM.

Chomicki, J.; and Imieliński, T. 1989. Relational Specifications of Infinite Query Answers. *ACM SIGMOD Record* 18(2): 174–183.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and Expressive Power of Logic Programming. *ACM Computing Surveys* 33(3): 374–425.

Das, A.; Gandhi, S. M.; and Zaniolo, C. 2018. ASTRO: A Datalog System for Advanced Stream Reasoning. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1863–1866. ACM.

Koutras, C. D.; and Nomikos, C. 2000. On the Computational Complexity of Stratified Negation in Linear-Time Temporal Logic Programming. *Intensional Programming II, World Scientific* 106–116.

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

Nomikos, C.; Rondogiannis, P.; and Gergatsoulis, M. 2005. Temporal Stratification Tests for Linear and Branching-Time Deductive Databases. *Theoretical Computer Science* 342(2–3): 382–415.

Przymusiński, T. C. 1988. On the Declarative Semantics of Deductive Databases and Logic Programs. In *Foundations of Deductive Databases and Logic Programming*, 193–216. Elsevier.

Ryzhikov, V.; Wałęga, P.; and Zakharyaschev, M. 2020. Temporal Ontology-Mediated Queries and First-Order Rewritability: A Short Course. In *Reasoning Web. Declarative Artificial Intelligence*, 109–148. Berlin: Springer.

Ryzhikov, V.; Wałęga, P. A.; and Zakharyaschev, M. 2019. Data Complexity and Rewritability of Ontology-Mediated Queries in Metric Temporal Logic under the Event-Based Semantics. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 1851–1857. IJCAI.

Wałega, P. A.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2020. Tractable Fragments of Datalog with Metric Temporal Operators. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 1919–1925. IJCAI.

Wałęga, P.; Cuenca Grau, B.; and Kaminski, M. 2019. Reasoning over Streaming Data in Metric Temporal Datalog. In *The 33rd AAAI Conference on Artificial Intelligence*, 1941–1948. AAAI Press.

Wałęga, P.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2020. DatalogMTL over Integer Timeline. In *Proceedings of the 17th International Conference on the Principles of Knowledge Representation and Reasoning*, 526–541. AAAI Press.

Wałęga, P. A.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2019. DatalogMTL: Computational Complexity and Expressive Power. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 1886–1892. IJCAI.

Zaniolo, C. 2012. Logical Foundations of Continuous Query Languages for Data Streams. In *Proceedings of the 2nd International Workshop on Datalog in Academia and Industry*, 177–189. Springer.