DatalogMTL: Computational Complexity and Expressive Power

Przemysław A. Wałęga[1,2], **Bernardo Cuenca Grau**[1], **Mark Kaminski**[1] and **Egor V. Kostylev**[1]
[1]Department of Computer Science, University of Oxford, UK
[2]Institute of Philosophy, University of Warsaw, Poland
{przemyslaw.walega, bernardo.cuenca.grau, mark.kaminski, egor.kostylev}@cs.ox.ac.uk

Abstract

We study the complexity and expressive power of DatalogMTL—a knowledge representation language that extends Datalog with operators from metric temporal logic (MTL) and which has found applications in ontology-based data access and stream reasoning. We establish tight PSpace data complexity bounds and also show that DatalogMTL extended with negation on input predicates can express all queries in PSpace; this implies that MTL operators add significant expressive power to Datalog. Furthermore, we provide tight combined complexity bounds for the forward-propagating fragment of DatalogMTL, which was proposed in the context of stream reasoning, and show that it is possible to express all PSpace queries in the fragment extended with the falsum predicate.

1 Introduction

DatalogMTL is an extension of Datalog with metric temporal operators that has been studied in the context of querying temporal data [Brandt *et al.*, 2017; Brandt *et al.*, 2018] and reasoning over data streams [Wałęga *et al.*, 2019].

DatalogMTL allows for MTL expressions in rules such as $\boxminus_{[k_1,k_2]}\varphi$ and $\diamondminus_{[k_1,k_2]}\varphi$, with k_1 and k_2 rational numbers, which hold at time t if φ holds at each and some, respectively, moment in the time interval $[t-k_2, t-k_1]$. Rules with such expressions can be used to naturally capture interesting events in temporal data. For instance, an analyst monitoring equipment data might be interested in flagging risky overheating events—that is, complex events where the temperature of a piece of equipment (1) has reached 60 Celsius, (2) has continuously exceeded 40 Celsius for a period of 10 seconds, or (3) has continuously exceeded 40 Celsius for two consecutive periods of 5 seconds with at most 2 seconds in-between. Such events are naturally captured by DatalogMTL rules (1)–(3), where numbers represent seconds:

$$\mathsf{Overheat}(x) \leftarrow \diamondminus_{[0,0]}\mathsf{TempAbove60}(x), \tag{1}$$

$$\mathsf{Overheat}(x) \leftarrow \boxminus_{[0,10]}\mathsf{TempAbove40}(x), \tag{2}$$

$$\mathsf{Overheat}(x) \leftarrow \boxminus_{[0,5]}\mathsf{TempAbove40}(x) \wedge \diamondminus_{[5,7]}\boxminus_{[0,5]}\mathsf{TempAbove40}(x). \tag{3}$$

Fact entailment in DatalogMTL has been recently studied by Brandt *et al.* (2018), who showed that it is ExpSpace-complete in combined and P-hard in data complexity, assuming binary encoding of numbers. Brandt *et al.* (2018) also studied the non-recursive fragment of DatalogMTL and showed that fact entailment becomes PSpace-complete and in P, respectively. Wałęga *et al.* (2019) defined forward-propagating DatalogMTL, which disallows punctual intervals of the form $[t, t]$ and imposes further syntactic restrictions to ensure that rule application cannot propagate derived information to the past; they also showed that fact entailment for this fragment is PSpace-complete in data complexity assuming binary encoding of numbers and P-complete assuming unary encoding. The results in [Brandt *et al.*, 2018] and [Wałęga *et al.*, 2019] place the data complexity of DatalogMTL in-between PSpace and ExpSpace; to the best of our knowledge, however, the exact data complexity of DatalogMTL has remained an open problem until now.

In this paper, we bridge this gap and show that the data complexity of fact entailment in DatalogMTL is in PSpace (and hence PSpace-complete). Similarly to [Brandt *et al.*, 2018], we assume binary encoding of numbers, and consider rules equipped with the MTL operators $\diamondminus_\varrho$, $\boxminus_\varrho$, and $\mathcal{S}_\varrho$, which refer to the past, and $\diamondplus_\varrho$, $\boxplus_\varrho$, and $\mathcal{U}_\varrho$, which refer to the future. To obtain the upper bound, we define in Section 3 two non-deterministic generalised Büchi automata relative to a DatalogMTL program and a temporal dataset, such that the languages of these automata are both non-empty if and only if the program and the dataset admit a common model. We then show that the emptyness condition for such automata can be checked using only polynomial space.

In Section 4 we revisit fact entailment in the forward-propagating fragment of DatalogMTL. Our results in Section 3 immediately extend the PSpace upper bound in data complexity by Wałęga *et al.* (2019) to programs equipped with the falsum predicate $\bot$ and punctual intervals of the form $[t, t]$. Concerning combined complexity, we establish a novel ExpSpace lower bound applicable to programs equipped with $\bot$ and an ExpTime upper bound (matching the complexity of plain Datalog) for programs without $\bot$.

Finally, we discuss the expressive power of DatalogMTL in the sense of descriptive complexity [Immerman, 1999]. We show that, under standard minor modifications to the language [Dantsin *et al.*, 2001], the forward-propagating frag-

$\mathfrak{M}, t \models \top$	for all $t \in \mathbb{Q}$
$\mathfrak{M}, t \not\models \bot$	for all $t \in \mathbb{Q}$
$\mathfrak{M}, t \models \diamondminus_\varrho A$	if $\mathfrak{M}, s \models A$ for some s with $t - s \in \varrho$
$\mathfrak{M}, t \models \boxminus_\varrho A$	if $\mathfrak{M}, s \models A$ for some s with $s - t \in \varrho$
$\mathfrak{M}, t \models \boxminus_\varrho A$	if $\mathfrak{M}, s \models A$ for all s with $t - s \in \varrho$
$\mathfrak{M}, t \models \diamondminus_\varrho A$	if $\mathfrak{M}, s \models A$ for all s with $s - t \in \varrho$
$\mathfrak{M}, t \models A\mathcal{S}_\varrho A'$	if $\mathfrak{M}, s \models A'$ for some s with $t - s \in \varrho$ and $\mathfrak{M}, r \models A$ for all $r \in (s, t)$
$\mathfrak{M}, t \models A\mathcal{U}_\varrho A'$	if $\mathfrak{M}, s \models A'$ for some s with $s - t \in \varrho$ and $\mathfrak{M}, r \models A$ for all $r \in (t, s)$

Table 1: Semantics of DatalogMTL ground literals

ment equipped with $\bot$ captures (i.e., can express all queries in) PSPACE. Together with the data complexity results in Section 3, this implies that DatalogMTL also captures PSPACE.

2 Preliminaries

Intervals. We consider intervals over $\mathbb{Q}$, denoted by $\langle x, y\rangle$, where $x, y \in \mathbb{Q} \cup \{-\infty, +\infty\}$, $\langle$ is [or (, $\rangle$ is] or), and

$$\langle x, y\rangle = \{t \in \mathbb{Q} \mid x \le t \le y \text{ and } t \ne x \text{ if } \langle \text{ is (, whereas } t \ne y \text{ if } \rangle \text{ is) }\}.$$

The left and right endpoints of an interval ϱ are denoted by ϱ^- and ϱ^+, respectively. An interval ϱ with $\varrho^- \ge 0$ is *positive* and an interval of the form $[t, t]$ is *punctual*. The length of an interval ϱ is denoted by $|\varrho|$. For non-empty intervals ϱ_1, ϱ_2, and a non-empty positive interval ϱ_3 we consider operations $\varrho_1 \cup \varrho_2$, $\varrho_1 \cap \varrho_2$, ϱ_1^c, $\varrho_1 - \varrho_3$, $\varrho_1 \ominus \varrho_3$, $\varrho_1 + \varrho_3$, and $\varrho_1 \oplus \varrho_3$, the result of each of which is an interval; the first two are the usual union and intersection and the others are defined in Table 2 (note that $\ominus$ and $\oplus$ can result in an empty interval, e.g., $[0, 1] \ominus [0, 2] = [0, -1]$). We assume that each $t \in \mathbb{Q}$ is represented by an integer numerator and a positive integer denominator, both encoded in binary.

Syntax of DatalogMTL. Assume a function-free first-order vocabulary with constants and predicates with non-negative integer arities. An *atom* is of the form $P(\boldsymbol{\tau})$ where P is a predicate and $\boldsymbol{\tau}$ is a matching arity tuple of constants and variables. A *literal* is an expression from the following grammar, for α an atom and ϱ a non-empty positive interval:

$$A{:=}\alpha \mid \top \mid \bot \mid \diamondminus_\varrho A \mid \boxminus_\varrho A \mid \boxminus_\varrho A \mid \diamondminus_\varrho A \mid A\mathcal{S}_\varrho A \mid A\mathcal{U}_\varrho A.$$

A (DatalogMTL) *rule* is an expression of the form:

$$B \leftarrow A_1 \wedge \cdots \wedge A_n, \tag{4}$$

where $n \ge 0$, each A_i is a literal, and B is a literal not mentioning operators $\diamondminus$, $\boxminus$, $\mathcal{S}$, and $\mathcal{U}$. A rule of the form (4) is *safe* if each variable in B occurs in some A_i. A (DatalogMTL) *program* is a finite set of safe rules [Brandt *et al.*, 2018]. The *greatest common divisor* $\mathsf{gcd}(\Pi)$ of a program Π is the greatest rational number which divides all rational numbers which are endpoints of intervals in Π to integer values (if Π has no numbers, we take $\mathsf{gcd}(\Pi) = 1$ for definiteness). An expression e (e.g., an atom, literal, or rule) is *ground* if it mentions no variables. An assignment ν of variables to constants is a *grounding* of e if $e\nu$ is ground. A (*temporal*) *fact* is an expression of the form $\alpha@\varrho$ with α a ground atom and ϱ a non-empty interval; a *generalised fact* is of the form $A@\varrho$, where A is a ground literal and ϱ a non-empty interval. A *dataset* is a finite set of facts. For a program Π and a dataset $\mathcal{D}$ we denote with $\mathsf{glit}(\Pi, \mathcal{D})$ the set of ground literals consisting of $\top$, all groundings of literals in Π by constants in Π and $\mathcal{D}$, and literals $\boxminus_{[0,+\infty)}\alpha$, $\boxminus_{(0,+\infty)}\alpha$, $\boxplus_{[0,+\infty)}\alpha$, and $\boxplus_{(0,+\infty)}\alpha$, for α an atom in $\mathcal{D}$.

Semantics of DatalogMTL. An *interpretation* $\mathfrak{M}$ specifies, for each ground atom α and each time point $t \in \mathbb{Q}$, whether α is true at t, in which case we write $\mathfrak{M}, t \models \alpha$. This notion extends to ground literals different from atoms as in Table 1. Interpretation $\mathfrak{M}$ is a *model* of a rule of the form (4) if, for each grounding ν and each $t \in \mathbb{Q}$, we have $\mathfrak{M}, t \models B\nu$ whenever $\mathfrak{M}, t \models A_i\nu$ for every $i = 1, \ldots, n$. Interpretation $\mathfrak{M}$ is a *model* of a program Π, written $\mathfrak{M} \models \Pi$, if $\mathfrak{M}$ is a model of all rules in Π. Interpretation $\mathfrak{M}$ is a *model* of a generalised fact $A@\varrho$ if $\mathfrak{M}, t \models A$ for all $t \in \varrho$. Interpretation $\mathfrak{M}$ is a *model* of a possibly infinite set $\mathcal{E}$ of generalised facts (e.g., a dataset), written $\mathfrak{M} \models \mathcal{E}$, if $\mathfrak{M}$ is a model of all the elements of $\mathcal{E}$. Note that each set of facts has the unique minimal model (i.e., a model assigning true as little as possible). A program Π and a possibly infinite set of generalised facts $\mathcal{E}$ are *consistent* if Π and $\mathcal{E}$ have a common model. A program Π and a set $\mathcal{E}$ *entail* a set of generalised facts $\mathcal{E}'$, written $(\Pi, \mathcal{E}) \models \mathcal{E}'$, if each model of Π and $\mathcal{E}$ is a model of $\mathcal{E}'$. If $\mathcal{E}'$ is a singleton $\{A@\varrho\}$, then we denote it simply by $A@\varrho$. Brandt *et al.* (2018) showed that fact entailment and consistency checking in DatalogMTL polynomially reduce to the complements of each other. Hence, in what follows we mostly concentrate on consistency and consider *combined complexity* of this problem when both program and dataset form the input, and *data complexity* when the program is fixed.

Normal Form. Similarly to Brandt *et al.* (2018), we usually assume that programs are in a normal form where there is no nesting of metric operators and where the form of intervals is restricted. For convenience in the automata construction later on, however, we refine their definition by further restricting the form of unbounded intervals. In particular, a program Π is in *normal form* if it contains only rules of the forms:

$$B \leftarrow \alpha_1 \wedge \cdots \wedge \alpha_n, \quad B \leftarrow \boxminus_\varrho \alpha, \quad B \leftarrow A\mathcal{S}_\varrho \alpha, \quad B \leftarrow \boxplus_\varrho \alpha, \quad B \leftarrow A\mathcal{U}_\varrho \alpha,$$

where each of $\alpha, \alpha_1, \ldots, \alpha_n$ is an atom, A is $\top$ or an atom, B is $\bot$ or an atom, and ϱ is a non-empty positive interval such that either it is bounded or it satisfies $\varrho^- = 0$.

Proposition 1. *Each program Π can be transformed in polynomial time into a program Π' in normal form such that $(\Pi, \mathcal{D}) \models \alpha@\varrho$ if and only if $(\Pi', \mathcal{D}) \models \alpha@\varrho$ for each dataset $\mathcal{D}$ and each fact $\alpha@\varrho$ over the vocabulary of Π.*

In what follows we assume that all programs are normal.

operation	left endpoint	right endpoint	left-open iff left endpoint is $-\infty$ or	right-open iff right endpoint is $+\infty$ or
ϱ_1^c	ϱ_1^-	ϱ_1^+	never	never
$\varrho_1 - \varrho_2$	$\varrho_1^- - \varrho_2^+$	$\varrho_1^+ - \varrho_2^-$	ϱ_1 is left-open or ϱ_2 is right-open	ϱ_1 is right-open or ϱ_2 is left-open
$\varrho_1 \ominus \varrho_2$	$\varrho_1^- - \varrho_2^-$	$\varrho_1^+ - \varrho_2^+$	ϱ_1 is left-open and ϱ_2 is left-closed	ϱ_1 is right-open and ϱ_2 is right-closed
$\varrho_1 + \varrho_2$	$\varrho_1^- + \varrho_2^-$	$\varrho_1^+ + \varrho_2^+$	ϱ_1 or ϱ_2 is left-open	ϱ_1 or ϱ_2 is right-open
$\varrho_1 \oplus \varrho_2$	$\varrho_1^- + \varrho_2^+$	$\varrho_1^+ + \varrho_2^-$	ϱ_1 is left-open and ϱ_2 is right-closed	ϱ_1 is right-open and ϱ_2 is left-closed

Table 2: Operations on intervals, where we assume that $-\infty + x = -\infty$ and $+\infty + x = +\infty$ for any value x (so $+$ is not commutative)

Canonical Interpretations. Consistency and fact entailment can be checked using the canonical interpretation of a program and a dataset, as defined next [Brandt *et al.*, 2018]. The immediate consequence operator T_Π of a program Π applied to a set $\mathcal{E}$ of generalised facts is the minimal set $T_\Pi(\mathcal{E})$ containing $\mathcal{E}$ and satisfying the following conditions, where Π' is the set of (results of) all groundings of rules in Π, and $\mathcal{E}'$ extends $\mathcal{E}$ with $\top@\varrho$ for each non-empty interval ϱ:

- **if** $B \leftarrow \alpha_1 \wedge \cdots \wedge \alpha_n$ is in Π', $\alpha_i@\varrho_i \in \mathcal{E}$ for $i = 1, \ldots, n$, and $\varrho = \bigcap_{i=1}^n \varrho_i$ is non-empty, **then** $B@\varrho \in T_\Pi(\mathcal{E})$;
- **if** $B \leftarrow \boxminus_{\varrho_2}\alpha$ is in Π', $\alpha@\varrho_1 \in \mathcal{E}$, and $\varrho = \varrho_1 \oplus \varrho_2$ is non-empty, **then** $B@\varrho \in T_\Pi(\mathcal{E})$;
- **if** $B \leftarrow \boxplus_{\varrho_2}\alpha$ is in Π', $\alpha@\varrho_1 \in \mathcal{E}$, and $\varrho = \varrho_1 \ominus \varrho_2$ is non-empty, **then** $B@\varrho \in T_\Pi(\mathcal{E})$;
- **if** $B \leftarrow A_1 \mathcal{S}_{\varrho_3} A_2$ is in Π', $A_i@\varrho_i \in \mathcal{E}'$ for $i = 1, 2$, and $\varrho = ((\varrho_1^c \cap \varrho_2) + \varrho_3) \cap \varrho_1^c$ is non-empty, **then** $B@\varrho \in T_\Pi(\mathcal{E})$;
- **if** $B \leftarrow A_1 \mathcal{U}_{\varrho_3} A_2$ is in Π', $A_i@\varrho_i \in \mathcal{E}'$ for $i = 1, 2$, and $\varrho = ((\varrho_1^c \cap \varrho_2) - \varrho_3) \cap \varrho_1^c$ is non-empty, **then** $B@\varrho \in T_\Pi(\mathcal{E})$;
- **if** $\alpha@\varrho_1 \in \mathcal{E}$, $\alpha@\varrho_2 \in \mathcal{E}$, and $\varrho = \varrho_1 \cup \varrho_2$ is non-empty, **then** $\alpha@\varrho \in T_\Pi(\mathcal{E})$.

The *canonical interpretation* $\Pi(\mathcal{D})$ of Π and $\mathcal{D}$ is the minimal model of all facts in $T_\Pi^{\omega_1}(\mathcal{D})$, with ω_1 the first uncountable ordinal, which is defined by application of T_Π as follows, for κ a successor ordinal and γ a limit ordinal:

$$T_\Pi^0(\mathcal{D}) = \mathcal{D},\ T_\Pi^{\kappa+1}(\mathcal{D}) = T_\Pi(T_\Pi^\kappa(\mathcal{D})),\ T_\Pi^\gamma(\mathcal{D}) = \bigcup_{\kappa<\gamma} T_\Pi^\kappa(\mathcal{D}).$$

Brandt *et al.* (2018) showed that if a program Π and a dataset $\mathcal{D}$ are consistent, then $\Pi(\mathcal{D})$ is their common model, and $\Pi(\mathcal{D})$ is a model of Π and $\mathcal{D}$ if and only if $\bot@\varrho \notin T_\Pi^{\omega_1}(\mathcal{D})$ for all ϱ; moreover, for every fact $\alpha@\varrho$, $(\Pi, \mathcal{D}) \models \alpha@\varrho$ if and only if $\Pi(\mathcal{D}) \models \alpha@\varrho$ or $\bot@\varrho' \in T_\Pi^{\omega_1}(\mathcal{D})$ for some ϱ'.

3 Data Complexity of DatalogMTL

In this section we establish our main result, namely that checking consistency and fact entailment in DatalogMTL is in PSPACE in data complexity. In the remainder of this section, we fix a program Π in normal form and a dataset $\mathcal{D}$.

In Section 3.1 we show that the canonical interpretation $\Pi(\mathcal{D})$ is structured according to regularly distributed consecutive intervals such that all points within an interval satisfy the same ground atoms. We refer to such intervals as *ruler-intervals* and to the set of their endpoints as the *ruler*. We also refer to any interpretation satisfying the same ground atoms on each ruler-interval as a *ruler-interpretation*.

In Section 3.2 we define two (infinite) generalised Büchi automata and show that their languages are both non-empty if and only if Π and $\mathcal{D}$ are consistent. Each state of the automata is a *window* over an interval, which corresponds to the projection of a ruler-interpretation into this interval. Each symbol in the alphabet is a set of ground literals in $\mathsf{glit}(\Pi, \mathcal{D})$. The initial states of both automata represent a window capturing the input dataset. Transitions between windows W_1 and W_2 are such that W_2 is the result of 'sliding' W_1 one step in the ruler. The sets of final states ensure that the ruler-interpretation obtained from the facts in the states of an accepting run satisfy also the generalised facts in those states.

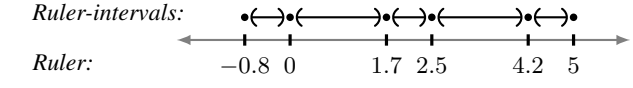
In Section 3.3 we show that non-emptyness of these automata can be checked using only polynomial space. This is non-trivial since the initial state can be of exponential size in the size of the dataset, and the number of states is unbounded. We address the first difficulty by guessing the initial state step-by-step. To address the second difficulty, we define equivalent automata with only exponentially many states, each of which is of polynomial size; these automata can be run 'on-the-fly' using only polynomial space.

3.1 Structure of the Canonical Interpretation

We next define the notions of a ruler, a ruler-interval, and a ruler-interpretation for Π and $\mathcal{D}$. Crucially, the time points in the ruler are regularly distributed and depend only on the time points in $\mathcal{D}$ and the greatest common divisor of Π.

Definition 2. The *ruler* (for Π and $\mathcal{D}$) is the set of time points of the form $x + n \cdot \mathsf{gcd}(\Pi)$ for x a rational in $\mathcal{D}$ (or 0 if $\mathcal{D} = \emptyset$) and n an integer. A *ruler-interval* is either a punctual interval over a time point on the ruler or an interval (t_1, t_2) with t_1, t_2 consecutive time points on the ruler. A ruler-interval ϱ' *precedes* or *succeeds* an interval ϱ with endpoints on the ruler if $\varrho' \cap \varrho$ is empty, $\varrho' \cup \varrho$ is a non-empty interval, and $(\varrho')^+ = \varrho^-$ or $(\varrho')^- = \varrho^+$, respectively. An interpretation $\mathfrak{M}$ is a *ruler-interpretation* if, for each ground atom α, we have that $\mathfrak{M}, t \models \alpha$ implies $\mathfrak{M}, t' \models \alpha$ for every time point t' in the same ruler-interval as t.

Example 3. *A fragment of the ruler and the corresponding ruler-intervals for a program* Π_{ex} *with* $\mathsf{gcd}(\Pi_{\text{ex}}) = 2.5$ *and a dataset* $\mathcal{D}_{\text{ex}} = \{\alpha@[0, 0], \beta@[4.2, 4.2]\}$ *is depicted below.*



Observe that the ruler contains all multiples of 2.5 *as well as all rationals of the form* $4.2 + 2.5 \cdot n$*; moreover,* $(4.2, 5)$ *succeeds* $[0, 4.2]$ *but does not succeed* $[0, 4.2)$.

The following lemma shows that the canonical interpretation is a ruler-interpretation, and hence we can concentrate only on ruler-interpretations when checking for consistency.

Lemma 4. *Interpretation $\Pi(\mathcal{D})$ is a ruler-interpretation.*

Proof sketch. It suffices to show by induction on the construction steps of $T_\Pi^{\omega_1}(\mathcal{D})$ that, for all $\alpha@\varrho \in T_\Pi^{\omega_1}(\mathcal{D})$, the endpoints ϱ^- and ϱ^+ are on the ruler. The base step of the induction trivially holds since $T_\Pi^0(\mathcal{D}) = \mathcal{D}$. For the inductive step, we can show that if $\alpha@\varrho$ is added by any of the rules in the definition of the consequence operator T_Π (see Section 2), then endpoints ϱ^- and ϱ^+ are of the required form. □

3.2 Büchi Automata for Consistency Checking

We start this section with the definition of windows, which play the role of states in the automata construction below.

Definition 5. Let ϱ be an interval with endpoints on the ruler. A *window* over ϱ is a set W of generalised facts $A@\varrho'$ with $A \in \mathsf{glit}(\Pi, \mathcal{D})$ and ruler-intervals $\varrho' \subseteq \varrho$ for which there exists a ruler-interpretation $\mathfrak{M}$, called *corresponding* to W, such that for all ruler-intervals $\varrho' \subseteq \varrho$ and $A \in \mathsf{glit}(\Pi, \mathcal{D})$ we have $\mathfrak{M} \models A@\varrho'$ if and only if $A@\varrho' \in W$.

Intuitively, a window W over ϱ provides a full description of a ruler-interpretation projected into ϱ; in particular, if $\mathfrak{M}$ is a ruler-interpretation corresponding to W, then $\mathfrak{M} \models W$.

Definition 6. A window W over an interval ϱ *satisfies* Π whenever, for each grounding $B \leftarrow A_1 \wedge \cdots \wedge A_n$ of a rule in Π and each ruler-interval $\varrho' \subseteq \varrho$, we have $B@\varrho' \in W$ if B is an atom and $A_i@\varrho' \in W$ for every $i = 1, \ldots, n$, and $A_i@\varrho' \notin W$ for some $i = 1, \ldots, n$ if $B = \bot$.

The following definitions about windows will also be useful. The *length* of a window over an interval ϱ is the number of ruler-intervals contained in ϱ (note that the length of the window is always a natural number, contrary to the length of ϱ). We denote by $\ell_{\min}$ the minimal length of a window over a closed interval ϱ with $|\varrho| = 2z$ for z the maximal rational in Π. Note that, since z is a multiple of $\mathsf{gcd}(\Pi)$, the window over $[x, x + 2z]$ has length $\ell_{\min}$ for each rational x on the ruler.

A window W_2 over an interval ϱ_2 *follows* a window W_1 over an interval ϱ_1 if the following holds:
- W_1 and W_2 have the same length;
- $\varrho_1 \cup \varrho_2 = \varrho_1' \cup (\varrho_1 \cap \varrho_2) \cup \varrho_2'$ for the ruler-intervals ϱ_1' and ϱ_2' preceding ϱ_2 and succeeding ϱ_1, respectively;
- $A@\varrho \in W_1$ if and only if $A@\varrho \in W_2$ for each ruler-interval $\varrho \subseteq \varrho_1 \cap \varrho_2$.

Intuitively, W_2 follows W_1 over ϱ_1 if W_2 is obtained from W_1 by deleting all generalised facts over the first ruler-interval in ϱ_1 and adding generalised facts over the ruler-interval succeeding ϱ_1. The next lemma establishes a useful closure property of windows that follow one another.

Lemma 7. *Let W_1 and W_2 be windows such that W_2 follows W_1. If W_1 and W_2 satisfy Π and are of length at least $\ell_{\min}$, then $W_1 \cup W_2$ is a window satisfying Π.*

We are ready to define the two automata that we use for consistency checking. The automata are parametrised by a window W, which determines the initial state.

Definition 8. For W a window of length ℓ satisfying Π, let $G_W^{\leftarrow}$ be the generalised Büchi automaton $(Q, \Sigma, \delta, W, \mathcal{F})$ with the following components:
- the states Q are all windows of length ℓ satisfying Π;
- the alphabet Σ is the powerset of $\mathsf{glit}(\Pi, \mathcal{D})$;
- the transition function δ is a partial function from $Q \times \Sigma$ to Q such that $\delta(W_1, \sigma) = W_2$ whenever W_2 is followed by W_1 and $W_2 \setminus W_1 = \{A@\varrho \mid A \in \sigma\}$;
- W is the initial state;
- accepting condition $\mathcal{F}$ (i.e., a set of sets of states) contains

$$\{W' \in Q \mid \boxminus_\varrho \alpha@\varrho' \in W' \text{ or } \alpha@\varrho' \notin W'\} \text{ and}$$
$$\{W' \in Q \mid \alpha \mathcal{S}_\varrho \beta@\varrho' \notin W' \text{ or } \beta@\varrho' \in W'\}$$

  for each $\boxminus_\varrho \alpha$ and each $\alpha \mathcal{S}_\varrho \beta$, respectively, in $\mathsf{glit}(\Pi, \mathcal{D})$ with unbounded ϱ, where ϱ' is any ruler-interval.

The automaton $G_W^{\rightarrow}$ is the same as $G_W^{\leftarrow}$ except that W_2 follows W_1 in the definition of δ, and $\boxplus$ and $\mathcal{S}$ are used instead of $\boxminus$ and $\mathcal{U}$, respectively, in $\mathcal{F}$.

The automata accept an infinite Σ-word $\sigma_1\sigma_2\ldots$ if there is a sequence $W_1, W_2, \ldots$ of states, called an *accepting run*, such that $W_1 = W$, $W_{i+1} = \delta(W_i, \sigma_i)$ for each $i \geq 1$, and the run has infinitely many states belonging to each set in $\mathcal{F}$.

Note that both automata in Definition 8 have infinitely many states; in the next section, we show that they are equivalent to finite generalised Büchi automata. Also, observe that the automata are essentially deterministic, except that certain pairs of a state and alphabet symbol are lacking a transition.

It would be useful to have a generalisation of Lemma 7 to an infinite number of windows. Although such a generalisation is not true in general, we next show that it holds for windows along accepting runs of the automata. We define an *infinite window* as in Definition 5 except that $\varrho = (-\infty, +\infty)$; all relevant notions transfer to infinite windows.

Lemma 9. *If W is a window of length at least $\ell_{\min}$ satisfying Π, then the union of all states in accepting runs of $G_W^{\leftarrow}$ and of $G_W^{\rightarrow}$ is an infinite window satisfying Π.*

Proof. Let W^* be the union of all states occurring in these accepting runs. Each state is a window satisfying Π, so W^* also satisfies Π. Let $\mathfrak{M}$ be the minimal model of the set of all facts in W^*. For each $A \in \mathsf{glit}(\Pi, \mathcal{D})$ with no unbounded interval and for each ruler-interval ϱ we have $\mathfrak{M} \models A@\varrho$ if and only if $A@\varrho \in W^*$; otherwise we would have a state in the accepting run without a corresponding ruler-interpretation (i.e., a state that is not a window). Since accepting runs visit each set in $\mathcal{F}$ infinitely often we can show that the above equivalence also holds for A with an unbounded interval. Thus, $\mathfrak{M}$ corresponds to W^*, and W^* is an infinite window. □

We are ready to show that automata $G_W^{\leftarrow}$ and $G_W^{\rightarrow}$ can be used to check whether Π and $\mathcal{D}$ are consistent. Let $\ell_\mathcal{D}$ be the length of a window over $[-x - z, x + z]$ for x the maximal absolute value of rationals in $\mathcal{D}$ and z the maximal rational in Π. Intuitively, $[-x - z, x + z]$ is big enough to contain all facts over bounded intervals from a dataset.

Theorem 10. *Program Π and dataset $\mathcal{D}$ are consistent iff there is a window W of length $\ell_\mathcal{D}$ satisfying Π such that the languages of both $G_W^{\leftarrow}$ and $G_W^{\rightarrow}$ are non-empty and $W \models \mathcal{D}$.*

Proof sketch. If Π and $\mathcal{D}$ are consistent, then they are satisfied by $\Pi(\mathcal{D})$, which is a ruler-interpretation by Lemma 4. For x the maximal absolute value of rationals in $\mathcal{D}$ and z the maximal rational in Π, the forward direction of the theorem is witnessed by the window W over $\varrho = [-x - z, x + z]$ such that, for each ground literal $A \in \mathsf{glit}(\Pi, \mathcal{D})$ and ruler-interval $\varrho' \subseteq \varrho$, $A@\varrho' \in W$ if and only if $\Pi(\mathcal{D}) \models A@\varrho'$.

Conversely, if W satisfies the conditions, then $G^{\leftarrow}_W$ and $G^{\rightarrow}_W$ have accepting runs. Let W^* be the union of all states in accepting runs of the automata, and $\mathfrak{M}$ the minimal model of facts in W^*. By the proof of Lemma 9, W^* is a window satisfying Π, and $\mathfrak{M}$ corresponds to W^*. Since $W \models \mathcal{D}$ we can show $\mathfrak{M} \models \mathcal{D}$; so, $\mathfrak{M}$ is a model of Π and $\mathcal{D}$. □

Although Theorem 10 suggests a consistency checking algorithm, windows of length $\ell_{\mathcal{D}}$ may contain an exponential (in the size of a dataset) number of elements; moreover, the number of states in the automata is unbounded. In the following section we show how to overcome these difficulties.

3.3 Complexity of Consistency Checking

We start by showing that we can decide whether a set of generalised facts is a window by guessing a polynomial representation of a corresponding interpretation.

Lemma 11. *Checking whether a finite set W of generalised facts is a window for program Π and dataset $\mathcal{D}$ is in* NP *in the size of (the representation of) $\mathcal{D}$.*

Proof sketch. If W is a window over ϱ, then it has a corresponding interpretation $\mathfrak{M}$ (minimally) witnessing the generalised facts in W. Such $\mathfrak{M}$ can be represented concisely as a set of facts such that if $\alpha@\varrho_1 \in W$ then $\alpha@\varrho_1 \in \mathfrak{M}$, if $\boxminus_{\varrho_2}\alpha@\varrho_1 \in W$ then $\alpha@(\varrho_1 - \varrho_2) \in \mathfrak{M}$, and if $\boxplus_{\varrho_2}\alpha@\varrho_1 \in W$ then $\alpha@(\varrho_1 + \varrho_2) \in \mathfrak{M}$. Moreover, for a generalised fact in W with $\mathcal{S}$ or $\mathcal{U}$ we need to guess at most two witnessing facts and add them to $\mathfrak{M}$. Importantly, for witnesses outside $[\varrho^- - z, \varrho^+ + z]$, for z the maximal rational in Π, the exact ruler-intervals in which they hold are irrelevant and only their order on the timeline needs to be kept, so the representation of $\mathfrak{M}$ can be guessed in NP. Checking whether $\mathfrak{M}$ is an interpretation corresponding to W can be clearly done in P. □

As a result, checking $W \models \mathcal{D}$ is in coNP for a window W, since we can guess a representation of a ruler-interpretation corresponding to W that does not satisfy $\mathcal{D}$. Note also that checking whether a window satisfies Π can be done in P.

A window W' over ϱ' is a *prefix* of W over ϱ if $\varrho' \subseteq \varrho$, the first ruler-intervals contained in ϱ and ϱ' are the same, and W' coincides with W on ϱ'; W' is a *suffix* of W if the same holds when considering the last contained ruler-intervals.

Lemma 12. *Let W be a window over interval ϱ of length $\ell \geq \ell_{\min}$ satisfying Π. Then, the prefix W' and the suffix W'' of W of length $\ell_{\min}$ are windows satisfying Π such that the languages of $G^{\leftarrow}_W$ and $G^{\leftarrow}_{W'}$ coincide, and the languages of $G^{\rightarrow}_W$ and $G^{\rightarrow}_{W''}$ coincide.*

Proof sketch. Words accepted by $G^{\leftarrow}_W$ are clearly accepted by $G^{\leftarrow}_{W'}$. For the converse, let w by accepted by $G^{\leftarrow}_{W'}$ and let W^* be the union of states in the run of $G^{\leftarrow}_{W'}$ on w. Then, as in Lemma 9, we can show that the ruler-interpretation corresponding to W^* is a model of Π, and it is possible to split W^* into a sequence of windows of length ℓ that form an accepting run of $G^{\leftarrow}_W$ on w. The case of W'' is symmetric. □

We are ready to show our main result.

Theorem 13. *Consistency checking and fact entailment in DatalogMTL are* PSPACE*-complete in data complexity.*

Proof sketch. The lower bound is inherited from the forward-propagating fragment [Wałęga *et al.*, 2019] (see also Section 4). By Theorem 10, to check if Π and $\mathcal{D}$ are consistent we can guess a set W of generalised facts and verify that W is a window of length $\ell_{\mathcal{D}}$ satisfying Π such that $W \models \mathcal{D}$ and the languages of $G^{\leftarrow}_W$ and $G^{\rightarrow}_W$ are non-empty. Although W may contain exponentially many elements, we can access it by guessing fragments of length $\ell_{\min}$ one-by-one. Then, by Lemma 7, it suffices to check whether each such fragment is a window satisfying Π; this can be done in NP by Lemma 11, which also implies that $W \models \mathcal{D}$ is in coNP.

For the non-emptiness checks, by Lemma 12 it suffices to verify, for the prefix W' and suffix W'' of W of length $\ell_{\min}$, whether the languages of $G^{\leftarrow}_{W'}$ and $G^{\rightarrow}_{W''}$ are non-empty. To overcome the problem with unbounded number of states, we obtain finite automata by merging states (i.e., windows) that are equivalent modulo a shift of the intervals over which the states are defined. Each of these finite automata is equivalent to its infinite version and has exponential (in the size of $\mathcal{D}$) number of states. Hence we can check non-emptiness by the standard 'on-the-fly' PSPACE procedure. □

4 Forward-Propagating Fragments

In this section we revisit the forward-propagating fragment of DatalogMTL proposed by Wałęga *et al.* (2019). In this fragment, rules are syntactically restricted to allow for derivation of facts into present and future time points, but precluding propagation towards past (in fact, we consider a slight extension of the language of Wałęga *et al.*, since we allow for $\bot$).

Definition 14. A DatalogMTL rule of the form (4) is *forward-propagating* if literal B does not mention $\boxminus$, and literals A_i do not mention $\diamondplus$, $\boxplus$, and $\mathcal{U}$. A program is *forward-propagating* if so is each of its rules.

As follows from Theorem 13, fact entailment (and hence consistency checking) for forward-propagating programs is in PSPACE in data complexity. Moreover, Wałęga *et al.* (2019) showed a matching lower bound for fact entailment that holds even if $\bot$ and punctual intervals are disallowed (note that programs without $\bot$ are always consistent).

Corollary 15. *Fact entailment for forward-propagating programs is* PSPACE*-complete in data complexity and stays* PSPACE*-hard if $\bot$ and punctual intervals are disallowed.*

Brandt *et al.* (2018) showed that fact entailment in the full DatalogMTL language is EXPSPACE-complete in combined complexity. The construction in their lower bound proof, however, does not apply to forward-propagating programs as it uses both past and future temporal operators. We next show that the problem for forward-propagating DatalogMTL

is also EXPSPACE-hard in combined complexity (and hence EXPSPACE-complete). Our proof, however, crucially relies on the use of $\bot$ in rule heads; in fact, we can show that fact entailment becomes EXPTIME-complete (and hence no harder than for plain Datalog) for the language without $\bot$.

Theorem 16. *Fact entailment for forward-propagating programs is* EXPSPACE*-complete in combined complexity and stays* EXPSPACE*-hard if punctual intervals are disallowed. The problem is* EXPTIME*-complete if $\bot$ is disallowed.*

Proof sketch. The EXPSPACE lower bound is obtained by reduction of the halting problem for Turing machines using exponential space. Our construction is a modification of the PSPACE-hardness construction by Wałęga *et al.* (2019), which reduces halting of polynomial-space bounded Turing machines to fact entailment for fixed forward-propagating programs without $\bot$ in the heads. Since the program is no longer fixed, we can use varying arity to encode binary representations of the tape position numbers, and use $\bot$ to derive inconsistencies that correspond to input rejection on doubly exponential steps of computation. The EXPTIME lower bound for programs without $\bot$ is a consequence of EXPTIME-hardness of fact entailment in plain Datalog; and the EXPTIME upper bound is obtained by analysis of the algorithm by Wałęga *et al.* (2019). Their algorithm for checking fact entailment derives facts only up to time t in the checked fact (given in binary). We can then show that the number of time points at which the algorithm derives facts is exponential in the value of t. In each of these time points at most exponentially many facts can be derived, and the derivation of all facts at a given time point is feasible in exponential time. □

5 Expressive Power

We finally discuss the expressivity of (the adapted in an appropriate way) DatalogMTL in the sense of descriptive complexity [Immerman, 1999] and start with relevant definitions.

A *logical language* is a set of sentences that evaluate on a finite set of ground atoms over *input predicates* to either true or false; as usual, we also assume that the input is *ordered*—that is, the set always contains atoms $first(c_1), next(c_1, c_2), \dots, next(c_{n-1}, c_n), last(c_n)$ for an enumeration $c_1, \dots, c_n$ of all constants in the set. Such a language *captures* a complexity class C if evaluating every fixed sentence in the language is in C, and for each problem in C there is a sentence in the language that is true if and only if its input encodes an instance of the problem. Under these definitions, DatalogMTL and its fragments are not logical languages, because atoms in their inputs have temporal labels and their programs do not evaluate to Boolean values. However, we can overcome these mismatches by adapting DatalogMTL to a logical language DatalogMTL$_{ll}$ as follows:

1. each A_i in a rule (4) may also be an atom over an input predicate or the negation of such an atom;
2. B in a rule (4) may also be the special nullary atom $Goal$, representing the positive answer.

Note that input predicates are not allowed in B and in complex literals in A_i. The semantics is extended to DatalogMTL$_{ll}$ programs on ordered inputs (i.e., sets of ground atoms over input predicates including the ordering atoms) in the standard way [Dantsin *et al.*, 2001], assuming that the input atoms hold at time point 0. It is not difficult to modify the PSPACE-completeness (i.e., both directions of Corollary 15) proof to show the following theorem.

Theorem 17. *DatalogMTL$_{ll}$ and forward-propagating fragment of DatalogMTL$_{ll}$ capture* PSPACE *over ordered inputs.*

6 Related Work

MTL is a prominent formalism for specifying and reasoning over complex events in real-time systems. MTL is equipped with two alternative semantics: pointwise and continuous, where the latter is adopted in this paper. Under the pointwise semantics, an interpretation is seen as a time-increasing sequence of pairs consisting of a time point and a set of propositional variables holding at that time point. In contrast, interpretations under the continuous semantics are functions mapping each element of the (dense) temporal domain to a set of propostional variables. Satisfiability and model checking of unrestricted MTL formulas are undecidable under the continuous semantics [Alur and Henzinger, 1993], but become decidable (EXPSPACE-complete) if punctual intervals are disallowed [Alur *et al.*, 1996].

DatalogMTL is a Horn fragment of MTL that extends the fundamental rule-based Datalog language [Abiteboul *et al.*, 1995; Dantsin *et al.*, 2001]. It was first studied under the continuous semantics in [Brandt *et al.*, 2017; Brandt *et al.*, 2018]. The non-recursive and forward-propagating fragments were studied in [Brandt *et al.*, 2018] and [Wałęga *et al.*, 2019] respectively. Low complexity fragments under the alternative pointwise semantics were investigated in [Kikot *et al.*, 2018].

DatalogMTL has been proposed as a suitable alternative to existing temporal languages for ontology-based data access [Artale *et al.*, 2017]. It has also found applications in the area of stream reasoning [Wałęga *et al.*, 2019], building on a long tradition of applications of MTL model checking to monitoring problems over data streams [Basin *et al.*, 2018; Baldor and Niu, 2012; Thati and Roşu, 2005; Doherty *et al.*, 2009; Ničković and Piterman, 2010; Ho *et al.*, 2014].

7 Conclusions

In this paper we have closed the existing gaps in the complexity of DatalogMTL and its forward-propagating fragment under the continuous semantics.

We see several avenues for future work. First, it would be interesting to investigate fragments of DatalogMTL with tractable fact entailment—an important requirement for data-intensive applications. Although Brandt *et al.* (2018) showed that reasoning in the non-recursive fragment is in AC^0, recursion remains an important feature in many applications. Second, it would be interesting to investigate the complexity of DatalogMTL under the alternative point-based semantics.

Acknowledgments

Research supported by the SIRIUS Centre for Scalable Data Access, the EPSRC projects DBOnto, MaSI^3, and ED^3, the NCN grant 2016/23/N/HS1/02168, and the Foundation for Polish Science (FNP).

References

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

[Alur and Henzinger, 1993] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Inf. Comput.*, 104(1):35–77, 1993.

[Alur *et al.*, 1996] Rajeev Alur, Tomás Feder, and Thomas A Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.

[Artale *et al.*, 2017] Alessandro Artale, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. Ontology-mediated query answering over temporal data: A survey (invited talk). In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 90, 2017.

[Baldor and Niu, 2012] Kevin Baldor and Jianwei Niu. Monitoring dense-time, continuous-semantics, metric temporal logic. In *RV*, pages 245–259, 2012.

[Basin *et al.*, 2018] David Basin, Felix Klaedtke, and Eugen Zălinescu. Algorithms for monitoring real-time properties. *Acta Inform.*, 55(4):309–338, 2018.

[Brandt *et al.*, 2017] Sebastian Brandt, R Kontchakov, V Ryzhikov, G Xiao, and M Zakharyaschev. Ontology-based data access with a Horn fragment of metric temporal logic. In *AAAI*, pages 1070–1076. AAAI Press, 2017.

[Brandt *et al.*, 2018] Sebastian Brandt, Elem Güzel Kalaycı, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. Querying log data with metric temporal logic. *J. Artif. Intell. Res.*, 62:829–877, 2018.

[Dantsin *et al.*, 2001] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.

[Doherty *et al.*, 2009] Patrick Doherty, Jonas Kvarnström, and Fredrik Heintz. A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems. *Auton. Agents Multi-Agent Syst.*, 19(3):332–377, 2009.

[Ho *et al.*, 2014] Hsi-Ming Ho, Joël Ouaknine, and James Worrell. Online monitoring of metric temporal logic. In *RV*, pages 178–192, 2014.

[Immerman, 1999] Neil Immerman. *Descriptive Complexity*. Springer, 1999.

[Kikot *et al.*, 2018] Stanislav Kikot, Vladislav Ryzhikov, Przemysław Andrzej Wałęga, and Michael Zakharyaschev. On the data complexity of ontology-mediated queries with MTL operators over timed words. In *DL*, 2018.

[Ničković and Piterman, 2010] Dejan Ničković and Nir Piterman. From MTL to deterministic timed automata. In *FORMATS*, pages 152–167, 2010.

[Thati and Roşu, 2005] Prasanna Thati and Grigore Roşu. Monitoring algorithms for metric temporal logic specifications. *Electron. Notes Theor. Comput. Sci.*, 113:145–162, 2005.

[Wałęga *et al.*, 2019] Przemysław Andrzej Wałęga, Bernardo Cuenca Grau, and Mark Kaminski. Reasoning over streaming data in metric temporal datalog. In *AAAI*, pages 1941–1948, 2019.