Tractable Fragments of Datalog with Metric Temporal Operators

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

Abstract

We study the data complexity of reasoning for fragments of DatalogMTL—an extension of Datalog with metric temporal operators over the rational numbers. Reasoning in DatalogMTL is PSPACE-complete, which handicaps its application in practice. To achieve tractability we first study the core fragment, which disallows conjunction in rule bodies, and show that reasoning remains PSPACE-hard. Intractability prompts us to also limit the kinds of temporal operators allowed in rules, and we propose a practical core fragment for which reasoning becomes \( \text{TC}^0 \)-complete. Finally, we show that this fragment can be extended by allowing linear conjunctions in rule bodies (with at most one IDB atom), and show that the resulting fragment is NP-complete, thus no harder than plain linear Datalog.

1 Introduction

DatalogMTL [Brandt et al., 2017; Brandt et al., 2018; Wałęga et al., 2019b] is a temporal extension of the fundamental rule language Datalog [Abiteboul et al., 1995] in which atoms in rules may contain metric temporal logic (MTL) operators interpreted over a rational timeline [Ouaknine and Worrell, 2008; Hunter et al., 2013]. For instance, DatalogMTL rule (1) states that cooling measures should be activated on a device \( x \) at time \( t \) if the device surpasses a fixed temperature threshold at some point in the interval \([t - 2.5, t]\); in contrast, rule (2) captures a more conservative requirement, where cooling measures are activated if the aforementioned threshold is exceeded continuously within the given time interval:

\[
\text{Cool}(x) \leftarrow \Theta_{[0,2.5]} \text{Temp}(x, \text{high}), \quad (1)
\]

\[
\text{Cool}(x) \leftarrow \exists_{[0,2.5]} \text{Temp}(x, \text{high}). \quad (2)
\]

MTL is equipped with two alternative semantics: pointwise and continuous; in this paper we consider the latter, which is the one typically adopted in works on DatalogMTL.

DatalogMTL is receiving increasing attention as a suitably expressive language for querying temporal data [Brandt et al., 2018; Artale et al., 2017; Ryzhikov et al., 2019] and reasoning over streams [Wałęga et al., 2019a]. The use of DatalogMTL in applications is, however, handicapped by its high data complexity of reasoning, namely PSPACE-complete for consistency checking [Wałęga et al., 2019b]. A known alternative to regain tractability while still allowing for temporal operators in rules is to preclude recursion altogether; Brandt et al. (2018) showed that, if timestamps and intervals are represented as dyadic rational numbers (rationals given as an integer fraction where the denominator is a power of 2), reasoning in nonrecursive DatalogMTL is first-order rewritable and hence \( \text{AC}^0 \) in data complexity.

In this paper, we study the data complexity of recursive fragments of DatalogMTL under continuous semantics. Our aim is to identify natural fragments with good computational properties and sufficient expressive power for data-intensive applications, while gaining a deeper understanding of interactions between constructs leading to intractability.

We first study the core fragment DatalogMTL\(_{\text{core}}\), where each rule with the head different from \( \bot \) has a singleton body; clearly, rules (1) and (2) satisfy this condition. Plain core Datalog and its extension with existential quantification in the rule heads are prominent KR languages, which can capture standard KR languages for ontology-based data access (OBDA) [Xiao et al., 2018; Motik et al., 2012]; reasoning in such languages is first-order rewritable, thus ensuring \( \text{AC}^0 \) data complexity [Call et al., 2012]. In Section 3 we show that this does not apply to our setting, as DatalogMTL\(_{\text{core}}\) remains PSPACE-hard in data complexity, even if the since operator \( S \) is the only temporal operator allowed in rules. Operator \( S \) fully expresses the past diamond operator \( \Box \) (but not vice-versa), and hence the rule (1) can be written using only \( S \) as a temporal operator; in contrast, \( S \) does allow us to express \( \exists \). This prompts us to investigate both DatalogMTL\(_{\Box}^{\text{core}}\)—the core fragment where \( \Box \) is the only temporal operator allowed—and the fragment DatalogMTL\(_{\exists}^{\text{core}}\). We show that reasoning in DatalogMTL\(_{\Box}^{\text{core}}\) is \( \text{TC}^0 \)-complete, and hence both tractable and parallelisable; however, these favourable computational properties do not transfer to DatalogMTL\(_{\exists}^{\text{core}}\) for which reasoning is P-hard.

In Section 4 we consider DatalogMTL\(_{\text{lin}}\)—the linear fragment, where rule bodies must contain at most one IDB conjunct (i.e., a conjunct whose predicate occurs in the head of some rule of the program). For instance, the program consist-

*Contact Author
ing of rule (3), which requires cooling measures to continue being applied if a moderate temperature is detected after cooling measures were already in force, is linear as the conjunction in (3) involves a single conjunct over IBD predicate Cool:

\[ \text{Cool}(x) \leftarrow \Phi_{[0,2]} \text{Cool}(x) \land \text{Temp}(x, \text{moderate}). \]  (3)

DatalogMTL_{lin} extends DatalogMTL_{core}, and hence all lower bounds proved in Section 4 transfer. To regain favourable computational properties, we focus on DatalogMTL_{lin}, the linear fragment where \( \Phi \) is the only temporal operator allowed. We then show that reasoning in this fragment is NL-complete, and hence no harder than in plain linear Datalog [Dantsin et al., 2001]. Linear Datalog provides the foundation for recursion in SQL, and hence our results open the door to efficient implementations of temporal recursive queries.

Our results are summarised in Figure 1 for reference (recall that TC^O \subseteq NL \subseteq P \subseteq PSPACE and NL \neq PSPACE).

2 Preliminaries

In this section, we define DatalogMTL and its associated reasoning problems [Brandt et al., 2018]. We also recapitulate some important properties of DatalogMTL proven by Walega et al. (2019b), which we exploit in our technical results.

Intervals. We consider intervals over rationals Q. An interval is denoted by \( \langle x, y \rangle \), where the left bracket (is \[ \text{or } \langle , \text{the right bracket} \rangle \) is \] or \), and \( x, y \in Q \cup \{ -\infty, +\infty \} \). Hence, \[ \langle x, y \rangle = \{ t \in Q \mid x \leq t \leq y, \] \( t \neq x \) if \( ( x, \text{and } t \neq y \) if \( ) \} \). \n
An interval \( \langle x, y \rangle \) is positive if \( x \geq 0 \); it is bounded if \( x, y \in Q \); and it is punctual if it is of the form \( [t, t] \), in which case we write \{ t \} instead of \( [t, t] \). Each number in Q is represented by an integer numerator and a positive integer denominator, both encoded in binary (we do not require rationals to be dyadic).

Syntax. Assume a function-free first-order vocabulary with constants and predicates equipped with a non-negative arity. An atom is of the form \( P(\tau) \), with \( P \) a predicate and \( \tau \) a tuple of constants and variables of matching arity. Each predicate in the vocabulary is either extensional (EDB) or intensional (IDB). A literal \( A \) is an expression given by the following grammar, with \( P(\tau) \) an atom and \( \phi \) a positive interval:

\[ A := P(\tau) \mid \top \mid \bot \mid \Phi \phi A \mid \Phi \phi A \mid \exists \phi A \mid \forall \phi A \mid AS\phi A \mid AU\phi A. \]

A literal is EDB if it mentions only EDB predicates and IDB otherwise. \( \text{(DatalogMTL rule is an expression of the form} \]

\[ B \leftarrow A_1 \land \cdots \land A_n, \quad \text{for } n \geq 0, \]

where each \( A_i \) is a literal and \( B \) is a literal not mentioning the operators \( \Phi, \Phi, \exists, \text{and } \forall. \) The conjunction of the \( A_i \) is the rule body, whereas the literal \( B \) is the rule head. A rule is safe if each variable in its head occurs also in its body. A DatalogMTL program is a finite set of safe rules. The greatest common divisor gcd(\( \Pi \)) of a program \( \Pi \) is the largest rational number dividing all rationals which are endpoints of intervals in \( \Pi \) to integer values (if \( \Pi \) has no numbers, we take \( \text{gcd}(\Pi) = 1 \) for definiteness). A program is core if each of its rules is of the form \( B \leftarrow A \) or \( \bot \leftarrow A_1 \land A_2 \). A program is linear if each of its rules is either of form (4) with at most one \( A_i \) being an IDB literal, or of the form \( \bot \leftarrow A_1 \land A_2 \). We denote the core and linear fragments by DatalogMTL_{core} and DatalogMTL_{lin}, respectively. For \( X \in \{ \Phi, \exists, \}, \) and \( Y \in \{ \text{core, lin} \}, \) we denote by DatalogMTL_{X} the fragment of DatalogMTL_{Y}, where \( X \) is the only temporal operator that may occur in literals. A program is propositional if all its predicates are nullary (i.e., propositions), and it is positive if it does not mention \( \bot. \) An expression \( e \) (e.g., an atom or a literal) is ground if it mentions no variables. A ground expression \( e' \) is a grounding of \( e \) if \( e' = e\nu \) for an assignment \( \nu \) of variables to constants. A fact is \( \alpha \circ \theta \), where \( \alpha \) is a ground atom and \( \theta \) is a non-empty interval. A dataset is a finite set of facts mentioning only EDB predicates.

Semantics. An interpretation \( \mathfrak{M} \) specifies, for each ground atom \( \alpha \) and each time point \( t \in Q \), whether \( \alpha \) is true at \( t \), in which case we write \( \mathfrak{M}, t \models \alpha. \) This notion extends to ground literals with temporal operators as in Table 1, where \( r, s, t \in Q. \) An interpretation \( \mathfrak{M} \) is a model of a rule of form (4) if, for each grounding assignment \( \nu \) and each \( t \in Q \), we have \( \mathfrak{M}, t \models B\nu \) whenever \( \mathfrak{M}, t \models A_i\nu \) for each \( i \in \{1, \ldots, n\}. \) An interpretation \( \mathfrak{M} \) is a model of program \( \Pi, \) written \( \mathfrak{M} \models \Pi, \) if \( \mathfrak{M} \) is a model of all rules in \( \Pi \). Interpretation \( \mathfrak{M} \) is a model of fact \( \alpha \circ \theta \), written \( \mathfrak{M} \models \alpha \circ \theta, \) if \( \mathfrak{M}, t \models \alpha \) for all time points \( t \in \theta \), and it is a model of dataset \( D, \) written \( \mathfrak{M} \models D, \) if it is a model of all facts in \( D. \) Program \( \Pi \) and dataset \( D \) are consistent if there is a model of both \( \Pi \) and \( D. \) Program \( \Pi \) and dataset \( D \) entail fact \( \alpha \circ \theta \), written \( (\Pi, D) \models \alpha \circ \theta, \) if \( \mathfrak{M} \models \alpha \circ \theta \) for each model \( \mathfrak{M} \) of \( \Pi \) and \( D. \) For brevity, we write \( D \models \alpha \circ \theta \) instead of \( (\emptyset, D) \models \alpha \circ \theta. \)

Reasoning Problems. We study the complexity of consistency checking for the linear and core fragments of DatalogMTL. We focus on data complexity—the standard measure of complexity in data-intensive applications, where programs are assumed to be fixed. All our complexity results are also applicable to the entailment of punctual facts of the form \( \alpha \circ \{ t \}, \) because such entailment and inconsistency checking problems are inter-reducible in all the fragments we consider. On the one hand, \( (\Pi, D) \models \alpha \circ \{ t \} \) if and only if \( \Pi \cup \{ \bot \leftarrow \alpha \land P \} \) and \( D \cup \{ P \circ \{ t \} \} \) are inconsistent, with
$\mathfrak{M}, t \models \top$ for all $t \in \mathbb{Q}$

$\mathfrak{M}, t \not\models \perp$ for all $t \in \mathbb{Q}$

$\mathfrak{M}, t \models \varnothing_\varrho A$ if $\mathfrak{M}, s \models A$ for some $s$ with $t - s \in \varrho$

$\mathfrak{M}, t \models \exists_\varrho A$ if $\mathfrak{M}, s \models A$ for some $s$ with $t - s \in \varrho$

$\mathfrak{M}, t \models \converse{\exists}_\varrho A$ if $\mathfrak{M}, s \models A$ for all $s$ with $t - s \in \varrho$

$\mathfrak{M}, t \models \Delta_\varrho A'$ if $\mathfrak{M}, s \models A'$ for some $s$ with $t - s \in \varrho$

$\mathfrak{M}, t \models \Delta_\varrho A'$ if $\mathfrak{M}, r \models A$ for all $r \in (t, s)$

<table>
<thead>
<tr>
<th>Table 1: Semantics of DatalogMTL ground literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$ a fresh proposition; on the other hand, $\Pi$ and $D$ are inconsistent if and only if $\Pi$ and $D$ entail a fact with a proposition not mentioned in $\Pi$ or $D$. In contrast, all known reductions of entailment of facts with arbitrary intervals to inconsistency checking require $S$ or $U$, which are not always available in our fragments. We assume without loss of generality that the least number occurring explicitly in each dataset is 0 (for a fixed program, any dataset can be modified to satisfy this assumption while preserving consistency).</td>
</tr>
</tbody>
</table>
|**Rulers.** If a DatalogMTL program $\Pi$ and a dataset $D$ are consistent, then they have a unique least model (i.e., the model assigning true as little as possible) [Brandt et al., 2017]. As shown by Wałęga et al. (2019b), such a least model can be divided into regularly distributed intervals whose time points satisfy the same ground atoms as follows. The $(\Pi, D)$-ruler is the set of time points of the form $x + n \cdot \gcd(\Pi)$, for $x$ a number in $\mathbb{N}$ and $n \in \mathbb{Z}$. A $(\Pi, D)$-interval is either a punctual interval over a time point on the ruler, or an interval $(t_1, t_2)$ with $t_1$ and $t_2$ consecutive time points on the ruler. For a sequence $\varrho_1, \varrho_1 + 1, \ldots, \varrho_j$ of successive $(\Pi, D)$-intervals (i.e., such that $\bigcup \varrho_i$ is an interval), with $i \leq j$, we define the (asymmetric) distances between $\varrho_i$ and $\varrho_j$ as $\text{dist}(\varrho_i, \varrho_j) = j - i$ and $\text{dist}(\varrho_i, \varrho_j) = -\text{dist}(\varrho_i, \varrho_j)$. As an example, we present below a fragment of the $(\Pi, D)$-ruler for $\Pi = \{\beta \leftarrow \varnothing_\{0, 2.5\}, \alpha \leftarrow \varnothing_\{5\}\}$ and $D = \{\alpha@t(0, 1.7)\}$, where $\gcd(\Pi) = 2.5$ and $\text{dist}(\{0\}, (2.5, 4.2)) = 5$.

$(\Pi, D)$-intervals: | (---) | (---) | (---) | (---) | (---) | (---) | (---) |

$(\Pi, D)$-ruler: | --- | 0 | -0.8 | 1.7 | 2.5 | 4.2 | 5 | --- |

The following lemma by Wałęga et al. (2019b) shows that points in a given ruler interval satisfy the same ground atoms.

**Lemma 1.** For every DatalogMTL program $\Pi$, dataset $D$, atom $\alpha$, and $t \in \mathbb{Q}$, if $(\Pi, D) \models \alpha@t$, then $(\Pi, D) \models \alpha@q$, where $q$ is the $(\Pi, D)$-interval such that $t \in q$.

The $(\Pi, D)$-ruler can be divided into segments of length $\gcd(\Pi)$ starting from 0, such that each segment contains the same number of $(\Pi, D)$-intervals and the corresponding intervals in different segments have the same lengths. The densit of $\Pi$ and $D$ is $\text{dist}(\{0\}, \{\gcd(\Pi)\})$; this induces an equivalence relation on $(\Pi, D)$-intervals with $d$ equivalence classes, which we call $(\Pi, D)$-types and denote by natural numbers between 0 and $d - 1$. The $(\Pi, D)$-type $k$ consists of all $(\Pi, D)$-intervals $\varrho$ such that $\text{dist}(\{0\}, \varrho) - k$ is a multiple of $d$.

### 3 Core Fragments

In this section, we study the data complexity of reasoning in the core DatalogMTL fragments. We first show that consistency checking in DatalogMTLcore is PSPACE-hard, which matches the known PSPACE upper bound for the full DatalogMTL language [Wałęga et al., 2019b]. In particular, each problem in PSPACE reduces to checking inconsistency of a dataset and a propositional DatalogMTLcore program where $S$ is the only temporal operator.

**Theorem 2.** Consistency checking in DatalogMTLcore is PSPACE-hard in data complexity. The result holds already for propositional programs using only $S$ as a temporal operator.

**Proof sketch.** Let $P$ be a problem in PSPACE. Then, there exists a deterministic Turing machine $M$ and a polynomial $p$ such that, for every input word $w$, $M$ decides $P$ for $w$ using at most $p(|w|)$ cells. We construct in a logarithmic space a dataset $D_w$ encoding the initial configuration of $M$ on $w$ and a propositional DatalogMTLcore program $\Pi_M$ simulating the computations of $M$ (on any input word) such that $M$ accepts $w$ if and only if $\Pi_M$ and $D_w$ are inconsistent.

Each configuration of $M$ in the run on $w$ is encoded within a unit interval (i.e., of length 1) divided into $p(|w|)$ subintervals, each describing a cell of the tape. Each subinterval is split into two (left-open and right-closed) halves indicated by propositions $P$ and $Q$, which respectively hold in the first and second half of each subinterval. In addition, we use the following propositions to encode a configuration: $P_a$—for each alphabet symbol $a$—which holds in the first half of a subinterval if the corresponding cell contains $a$; $Q_g$—for each state $g$—which holds in the second half of a subinterval if the head is over the corresponding cell and $M$ is in $g$; and $Q_L$ and $Q_R$, which hold in the second half of a subinterval if the corresponding cell is, respectively, somewhere to the left and to the right of the head’s position.

Program $\Pi_M$ has two main blocks. The first block consists of the following rules, for each symbol $a$ and state $q$:

$$P \leftarrow \top S_{\{1\}} P, \quad Q \leftarrow \top S_{\{1\}} Q,$$

$$Q_{aq} \leftarrow \top S_{\{1\}} (Q_{aq} S_{\{0, \infty\}} P_a), \quad P_{aq} \leftarrow \top S_{\{0, \infty\}} Q_{aq},$$

$$Q_{aq} \leftarrow \top S_{\{1\}} (Q_{aq} S_{\{0, \infty\}} P_a), \quad Q_{aq} \leftarrow \top S_{\{0, \infty\}} (P_{aq}).$$

The first two rules propagate $P$ and $Q$. The remaining rules copy the description of a configuration into the next unit interval using auxiliary propositions (note that there is a gap between the unit intervals encoding consecutive configurations). The rules ensure that, if $P_a$ and $Q_{aq}$ hold in the two halves of a subinterval, then $P_{aq}$ and $Q_a$ hold in the two halves of the corresponding subinterval in the next unit interval. Moreover, if $P_{aq}$ and $Q_{aq}$ hold in the same subinterval, then $Q_{aq}$ and $Q_{aq}$ hold in the second halves of the subintervals that are adjacent, respectively, to the left and to the right.

The second block of $\Pi_M$ encodes the transition function of $M$ (and does not use temporal operators). Each left-moving
transition \((q,a) \mapsto (q',a',L)\) is encoded by the rules
\[
P_{a'} \leftarrow P_{aq}, \quad Q_{a'} \leftarrow Q_{aq}^{L}, \quad Q_{R} \leftarrow Q_{aq}, \quad Q_{R} \leftarrow Q_{aq}^{R}.
\]
Right-moving transitions are encoded analogously; similar rules are also needed to propagate \(Q_{L}, Q_{R}, \) and \(P_{a'}\) for cells without the head. Finally, \(\Pi_M\) contains the rule \(\bot \leftarrow Q_{q}\) for each accepting state \(q\), which ensures inconsistency of \(\Pi_M\) and \(D_w\) when \(M\) accepts \(w\).

The since operator \(S\) is strictly more expressive than the past diamond operator \(\diamond\). Hence, to regain tractability, we consider the DatalogMTL\(\Phi\) fragment and show that, in this case, consistency checking is TC\(^0\)-complete. This suggests suitability of DatalogMTL\(\Phi\) for data-intensive applications.

We start by defining a normal form. A DatalogMTL\(\phi\) program is \emph{normal} if each of its rules has one of the following forms, where \(\alpha, \alpha_1, \alpha_2, \) and \(\beta\) are atoms, and \(\phi\) is a non-empty bounded positive interval:
\[
\beta \leftarrow \diamond\alpha, \quad \beta \leftarrow \top, \quad \bot \leftarrow \alpha_1 \land \alpha_2.
\]
Each DatalogMTL\(\phi\) program can be normalised so that the resulting program is a conservative extension of the original one: rules with nested operators are flattened by introducing fresh predicates, rules \(\alpha \leftarrow \alpha'\) without metric operators are rewritten as \(\alpha \leftarrow \diamond \{0\} \alpha'\), and rules \(P(\tau) \leftarrow \Theta_{[t,\infty)} P'(\tau')\) with unbounded intervals are replaced with rules \(Q(\tau) \leftarrow \Theta_{(t,1)} P'(\tau'), Q(\tau) \leftarrow \Theta_{[0,1]} Q(\tau),\) and \(P(\tau) \leftarrow \Theta_{[0,1]} Q(\tau)\), for \(Q\) a fresh predicate of the same arity as \(P\). Hence, in this section, we concentrate on normal DatalogMTL\(\phi\) programs.

Next, we provide a characterisation of fact entailment for ground normal DatalogMTL\(\phi\) programs in terms of existence of a specific length path in a graph defined below.

\textbf{Definition 3.} Let \(\Pi\) be a ground normal DatalogMTL\(\phi\) program and let \(d \in \mathbb{N}\). Then, \(G_{\Pi}^{d}\) is the directed edge-weighted multigraph with a vertex \(v_{\alpha}^d\) for each (ground) atom \(\alpha\) in \(\Pi\) and each \(k \in \{0,\ldots,d-1\}\); and an edge \((v_{\alpha}^d, v_{\beta}^d)\) of weight \(n\) whenever \(\ell = (k+n)\) mod \(d\) and \(\Pi\) has a rule \(\beta \leftarrow \diamond \phi \alpha\) such that \(n \cdot \gcd(\Pi/d) \in \phi\).

The graph \(G_{\Pi}^{d}\) is defined so that, if \(d\) is the density of \(\Pi\) and a dataset \(D\), then every derivation from \(\Pi\) and \(D\) (i.e., a sequence of actual facts) corresponds to a path in \(G_{\Pi}^{d}\) such that a 1-step derivation of \(\beta@\{t_2\}\) from \(\alpha@\{t_1\}\) corresponds to an edge \((v_{\alpha}^d, v_{\beta}^d)\) of weight \(\dist\(q_1, q_2\)\) where \(q_1 \equiv t_1\) and \(q_2 \equiv t_2\) are \((\Pi, D)\)-type intervals of \((\Pi, D)\)-types \(k\) and \(\ell\), respectively. Indeed, the following result holds.

\textbf{Lemma 4.} Let \(\Pi\) be a ground and positive normal DatalogMTL\(\phi\) program, \(D\) a dataset, and \(d\) the density of \(\Pi\) and \(D\). Then, for each fact \(\beta@g\) with a \((\Pi, D)\)-interval \(g\), the following are equivalent:
1. \((\Pi, D) \models \beta@g\);
2. there exist a fact \(\alpha@g'\) with \(g'\) a \((\Pi, D)\)-interval such that \(D \models \alpha@g'\) (or \(\Pi\) contains a rule \(\alpha \leftarrow \top\)) and a path in \(G_{\Pi}^{d}\) of weight \(\dist\(q_1, q_2\)\) from \(v_{\alpha}^d\) to \(v_{\beta}^d\), where \(k\) and \(\ell\) are the \((\Pi, D)\)-types of \(g'\) and \(g\), respectively.

We next show that, given two vertices \(v\) and \(v'\) in a directed multigraph with weights of edges being non-negative integers (e.g., \(G_{\Pi}^{d}\)), we can finitely represent the weights of all paths from \(v\) to \(v'\). This lemma relies on the normal form for unary non-deterministic finite automata (NFAs) by Chrobak (1986) and will be essential to prove the TC\(^0\) upper bound.

\textbf{Lemma 5.} Let \(G\) be a directed edge-weighted multigraph with non-negative integer weights and let \(v, v'\) be vertices in \(G\). Then, it is possible to construct a finite set \(S\) of pairs of non-negative integers such that there is a path in \(G\) of weight \(n\) from \(v\) to \(v'\) if and only if \(n = a + i \cdot b\) for a pair \((a, b)\) \(\in S\) and a non-negative integer \(i\).

\textbf{Proof sketch.} First, we replace each edge in \(G\) of weight \(n > 1\) with a chain of \(n\) edges of weight \(1\) by introducing fresh nodes. Then, we construct an NFA \(A\) over a unary alphabet \(\{\sigma\}\), where states correspond to vertices: the initial state corresponds to \(v\) and the final state to \(v'\), and we have a \(\sigma\)-transition for each edge of weight 1 and an \(e\)-transition for each edge of weight 0. Then, the weights of paths in \(G\) from \(v\) to \(v'\) are exactly the lengths of words accepted by \(A\). After eliminating \(e\)-transitions, we obtain a unary NFA, which we then transform to Chrobak normal form [Chrobak, 1986, Lemma 4.3], where the transitions form a path from the initial state followed by a single nondeterministic choice between several (disjoint) cycles. For every accepting state \(q\) in the resulting automaton \(A'\), there exist \(a, b \in \mathbb{N}\) such that there is an accepting run of \(A'\) on \(\sigma^n\) ending at \(q\) if and only if there is a number \(i\) with \(n = a + i \cdot b\) (where \(a\) and \(b\) are the lengths of the simple paths from the initial state to \(q\) and of the cycle through \(q\), respectively).

Lemmas 4 and 5 suggest a TC\(^0\) algorithm for checking whether a fixed ground positive normal DatalogMTL\(\phi\) program \(\Pi\) and a dataset \(D\) entail \(\beta@\{t\}\). For this, it suffices to check if there is \(\alpha@\Phi\) in \(D\) such that \((\Pi, \{\alpha@\Phi\}) \models \beta@\{t\}\). These checks can be done in parallel for each fact \(\alpha@\Phi\) in \(D\) and hence we prove that each of them is feasible in TC\(^0\). Assume that \(D' = \{\alpha@\Phi\}\); we want to check whether \((\Pi, D') \models \beta@\{t\}\). First, we compute the density \(d\) of \(\Pi\) and \(D'\), which is at most 4 since \(D'\) contains only one interval. Then, we compute the \((\Pi, D')\)-interval \(g'\) containing \(t\), and the \((\Pi, D')\)-type \(\ell\) of \(g'\). We also compute the left-most, \(g'^L\), and the right-most, \(g'^R\), \((\Pi, D')\)-intervals contained in \(g'\). All these computations are feasible in TC\(^0\). Now, by Lemma 4, it suffices to check whether there is a path of weight \(n\) from \(v_{g'^L}^d\) to \(v_{g'^R}^d\) in \(G_{\Pi}^{d}\) such that \(\dist(g'^L, g') \leq n \leq \dist(g'^R, g')\) and \(k \in K\) (if \(\Pi\) contains rules of the form \(\alpha' \leftarrow \top\), some additional but easy to handle checks are required), which by Lemma 5 can also be checked in TC\(^0\).

To check consistency of a ground normal DatalogMTL\(\phi\) program \(\Pi\) and a dataset \(D\), we need to consider rules of the from \(\bot \leftarrow \alpha_1 \land \alpha_2\), where \(\alpha_1\) and \(\alpha_2\) are ground atoms. In particular, inconsistency arises whenever there is a time point where both \(\alpha_1\) and \(\alpha_2\) hold together in the least model of \(D\) and the positive subset of \(\Pi\). This leads to technical difficulties since there is an unbounded number of candidate time points. The following theorem shows that these difficulties
can be overcome, and consistency checking is $TC^0$-complete, where the upper bound holds also for non-ground programs.

**Theorem 6.** Consistency checking for DatalogMTL_{core} is $TC^0$-complete in data complexity. The lower bound holds for propositional DatalogMTL_{core} without metric operators.

**Proof sketch.** For the lower bound, there is a simple $AC^0$ reduction of the integer multiplication problem [Hesse, 2001]: $a \cdot b = c$ for positive integers $a$, $b$, and $c$ (encoded in binary) if and only if the program $\{ \bot \leftarrow A \land B \}$, for propositions $A$ and $B$, and the dataset $\{ A \circ \{ a \}, B \circ \{ b \} \}$ is inconsistent.

For the upper bound, we first consider a ground normal DatalogMTL_{core} program $\Pi$, with $\Pi^+$ its positive subset. Since rules in $\Pi$ with $\bot$ do not contain temporal operators, the $(\Pi, D)$- and $(\Pi^+, D)$-rulers and the corresponding densities coincide, as well as $G_{\Pi}^D = G_{\Pi^+}^D$, for each $d \in \mathbb{N}$.

By linearity of derivations and by Lemma 1, $\Pi$ and $D$ are inconsistent if and only if there is a rule $\bot \leftarrow \beta_1 \land \beta_2$ in $\Pi$ and a subset $D' \subseteq D$ with at most two facts such that $(\Pi^+, D') = \beta_1 \circ \beta_2$ for both $i$ and some $(\Pi^+, D')$-interval $\varrho$. Consider such $D' = (\alpha_1 \circ \beta_1, \alpha_2 \circ \beta_2)$. Moreover, for both $i$, let $\varrho_i^L$ and $\varrho_i^R$ be the left- and right-most $(\Pi^+, D')$-intervals in $\varrho_i$, respectively, let $K_i$ be the sets of the $(\Pi^+, D')$-types of all $(\Pi^+, D')$-intervals in $\varrho_i$, and let $d$ be the density of $\Pi^+$ and $D'$. Then, assuming for simplicity that $\Pi$ does not have rules $\alpha \leftarrow \tau$, we use Lemma 4 to reduce existence of $p$ as above to existence of two paths in $G_{\Pi^+}^D$: one from $v_{k_1}^0$ to $v_{k_1}^1$ with weight $n_1$ and another from $v_{k_2}^0$ to $v_{k_2}^1$ with weight $n_2$, where $k_i \in K_i$ for both $i$, $\ell$ is a $(\Pi^+, D')$-type, and

$$\text{dist}(\varrho_i^L, \varrho_i^R) \leq n_1 - n_2 \leq \text{dist}(\varrho_i^L, \varrho_i^R).$$

(5)

We can check existence of such pairs of paths in parallel for each combination of $K_1, K_2$, and $\ell$. By Lemma 5, we can represent the weights of all paths from $v_{k_1}^0$ to $v_{k_1}^1$ by a finite set $S_i$ of integer pairs. So to check (5) it suffices to verify if for some $(a_1, b_1) \in S_1$ and $(a_2, b_2) \in S_2$ there exist non-negative integers $i_1, i_2$ with

$$\text{dist}(\varrho_i^L, \varrho_i^R) \leq (a_1 + i_1 \cdot b_1) - (a_2 + i_2 \cdot b_2) \leq \text{dist}(\varrho_i^L, \varrho_i^R),$$

which, by Bézout’s identity, exist if and only if the following interval contains an integer:

$$\left[ \frac{\text{dist}(\varrho_i^L, \varrho_i^R) - a_1 + a_2}{\gcd(b_1, b_2)}, \frac{\text{dist}(\varrho_i^L, \varrho_i^R) - a_1 + a_2}{\gcd(b_1, b_2)} \right].$$

The existence of such an integer can be verified in $TC^0$ (for all $(a_1, b_1) \in S_1$ and $(a_2, b_2) \in S_2$ in parallel); hence, we can check the consistency in $TC^0$ in the size of $D$.

Finally, let $\Pi$ be a non-ground normal DatalogMTL_{core} program. Note that although the grounding $\Pi_D$ of $\Pi$ with constants from $D$ can be computed in $AC^0$, the graph $G_{\Pi_D}^D$ depends not only on $\Pi$ and $D$ but also on $\Pi_D$; this would prevent us from exploiting the $TC^0$ procedure for ground programs. However, if $D'$ and $D''$ are the same modulo renaming of constants not occurring in $\Pi$, then $G_{\Pi_D}^{D'}$ and $G_{\Pi_D}^{D''}$ are isomorphic. Since we are interested in datasets with at most two facts and with arity of predicates fixed by $\Pi$, the number of non-isomorphic graphs for such datasets can be bounded without knowing $D$. Thus, we can compute the representatives of such non-isomorphic graphs in a data-independent way and reuse our $TC^0$ procedure for ground programs.

Now, we show P-hardness for consistency checking in DatalogMTL_{core} that holds even for propositional programs.

**Theorem 7.** Consistency checking in propositional DatalogMTL_{core} is P-hard in data complexity.

**Proof sketch.** The proof uses some ideas from the hardness proofs for pointwise semantics by Ryzhikov et al. (2019). We reduce the well-known P-complete path system accessibility problem (PSA) to inconsistency. An instance $G$ of PSA is a tuple $(V, E, S, v_s)$, where $V = \{ v_1, \ldots, v_n \}$ is a finite set of nodes, $E$ is a ternary relation on $V$ such that $i < j < k$ for each $(v_i, v_j, v_k) \in E$, $S \subseteq V$ is a set of (source) nodes, and $v_s \in V$ is a (target) node. The answer to $G$ is affirmative if $v_s$ is accessible, where a node $v_k$ is accessible if $v_k \in S$ or there are accessible nodes $v_i, v_j$ such that $(v_s, v_i, v_j, v_k) \in E$.

Let $\Pi$ be the (fixed) program consisting of the following rules over propositions $P, P', P''$, and $Q$:

$$P' \leftarrow \square \{ 2 \} P, \quad P'' \leftarrow \square \{ 0 \} P', \quad P \leftarrow \square \{ 1 \} P, \quad P \leftarrow \square \{ 1 \} P, \quad \bot \leftarrow P \land Q.$$

Consider now a PSA instance $G$ as above, and let $c_1, \ldots, c_m$ be an enumeration of $E$ preserving the ‘application order’ — that is, such that for every edge $e_t$ with node $v_k$ as the third argument, $v_k$ is not the first or second argument of any $e_t$ with $h < t$. Let $D_G$ be a dataset containing the following facts, for every $\ell$ such that $1 \leq \ell \leq m + 1$:

$$Q \circ \{ 4 \ell + t/n \} \quad \text{and} \quad Q \circ \{ 4 \ell + s/n \},$$

for every $v_s \in S$, and the following facts, for every $\ell$:

$$P' \circ \{ 4 \ell + 1 + k/n \} \quad P'' \circ \{ 4 \ell + 2 + j/n \}.$$

For instance, $D_G$, for $G$ with $V = \{ v_1, v_2, v_3 \}$, $E = \{ (v_1, v_2, v_3) \}$, $S = \{ v_1, v_2 \}$, and $t = 3$, is as follows.

by $\Pi$:

$$D_G: \quad P \quad Q \quad P' \quad P'' \quad P'$$

$$4 1/2 4 1/2 5 1/2 6 1/2 6 1/2 8 1/2 8 1/2 8 1/2$$

$(v_1, v_2, v_3) \in E$

It can then be easily checked that $v_t$ is accessible if and only if $\Pi$ and $D_G$ are inconsistent.

4 Linear Fragments

Since linear fragments of DatalogMTL are extensions of corresponding core fragments, all lower bounds established in Section 4 for core fragments hold also for linear fragments. To obtain favourable computational properties, we focus on DatalogMTL_{core} and show NL-completeness of consistency checking in data complexity. The lower bound is inherited
from plain linear Datalog, but we will prove it also for propositional DatalogMTL programs.

For the upper bound, we will concentrate on programs in a normal form, extending the one for the core case: a DatalogMTL program is normal if it consists only of rules of the following form, where \( \alpha, \alpha_1, \alpha_2 \) and \( \beta \) are IDB atoms, \( \phi \) is a possibly empty conjunction of EDB literals, and \( \varrho \) is a non-empty bounded positive interval:

\[
\beta \leftarrow \Phi \alpha \land \varphi,
\beta \leftarrow \top,
\bot \leftarrow \alpha_1 \land \alpha_2.
\]

Each DatalogMTL program can be normalised similarly as before—that is, by introducing rules with fresh predicates.

Naively, an inconsistency checking algorithm could guess a derivation of \( \bot \) from a normal program and an input dataset. This, however, does not lead to an NL algorithm since the length of a derivation and the representation of its elements (in particular numbers occurring in intervals) are unbounded.

We next show how to overcome this problem.

**Theorem 8.** Consistency checking for DatalogMTL is NL-complete in data complexity. The lower bound holds already for propositional DatalogMTL

**Proof sketch.** The lower bound is by reduction of consistency checking for propositional normal DatalogMTL under pointwise semantics, where datasets consist of punctual facts and only the time points explicit in the dataset constitute the temporal universe of interpretations; this problem is NL-complete in data complexity [Ryzhikov et al., 2019]. Given a propositional normal DatalogMTL program \( \Pi \), we introduce a fresh IDB proposition \( Q_A \) for each literal \( A \) of the form \( \Phi \alpha \) in \( \Pi \), and a fresh EDB proposition \( Q \). We then construct a propositional DatalogMTL program \( \Pi' \) by replacing each literal \( A \) as above in \( \Pi \) by \( Q_A \) and adding rules \( Q_A \leftarrow A \land Q \). For each (punctual) dataset \( D \), \( \Pi \) and \( D \) are consistent under pointwise semantics if and only if so are \( \Pi' \) and \( D \cup \{ Q \} \) under continuous semantics.

For the upper bound, consider a normal DatalogMTL program \( \Pi \) and a dataset \( D \). By Lemma 1, \( \Pi \) and \( D \) are inconsistent if and only if there are a grounding \( \bot \leftarrow \beta_1 \land \beta_2 \) of a rule in \( \Pi \) and derivations of both facts \( \beta_1 \land \varphi \) for some \( (\Pi, D, \varphi) \). We can see each derivation as a sequence of facts over ruler intervals with IDB predicates. Since, such a sequence may be of exponential length, to achieve the bound, we will represent such derivations concisely as follows. Let \( \Phi \) be the set of all groundings of the conjunctions \( \phi \) of EDB literals in rules of \( \Pi \) such that all the atoms in the groundings are in \( D \). For each \( \phi \in \Phi \), we compute the maximal intervals in which \( D \) entails \( \phi \), which is feasible in \( \text{TC}^0 \). Such intervals allow us to compute in \( L \) the partitioning of the whole \( \mathbb{Q} \) into sections—that is, the maximal intervals such that if \( D \) entails \( \phi \) in some time point in a section \( \sigma \), then \( D \) entails \( \phi \) in the whole \( \sigma \). Let \( \Phi_\sigma \) be the set of all \( \phi \in \Phi \) entailed in a section \( \sigma \) by \( D \). We represent each derivation sequence as the subsequence of polynomial length containing only the first and the last facts in the derivation from every section. Each interval \( \varrho \) in the subsequence that may have endpoints with exponentially long binary representations is represented concisely as a pair of a pointer to the time point \( t \) from \( D \) closest to \( \varrho \) and the number \( \text{dist}(\{ t \}, \varrho) \), which is polynomially representable in the size of \( D \) (special care is required when \( \varrho \) is in the first/last section). Hence, such representation of a derivation can be guessed fact by fact in NL.

Next, we show how to verify correctness of a guessed representation. The key step is to check that each fact \( \alpha_{i+1} \land \varrho_{i+1} \) in the subsequence can be derived from the preceding fact \( \alpha_i \land \varrho_i \). If \( \varrho_i \) and \( \varrho_{i+1} \) are in different sections, then we check, in \( L \), whether there is a 1-step derivation of \( \alpha_{i+1} \land \varrho_{i+1} \) from \( \alpha_i \land \varrho_i \). Otherwise—that is, when \( \varrho_i \) and \( \varrho_{i+1} \) are in the same section \( \sigma \)—the derivation of \( \alpha_{i+1} \land \varrho_{i+1} \) from \( \alpha_i \land \varrho_i \) can be of exponential length. To deal with this, we construct in \( L \) a program \( \Pi_\sigma \) by first deleting each rule in the grounding of \( \Pi \) mentioning some EDB conjunction not in \( \Phi_\sigma \), and then deleting all EDB literals from the remaining rules. Then \( \Pi_\sigma \) is a ground normal DatalogMTL program, so we can show, as in the proof of Lemma 4, that the following are equivalent:

1. there is a derivation using \( \Pi \) and \( D \) that starts with \( \alpha_i \land \varrho_i \) and ends with \( \alpha_{i+1} \land \varrho_{i+1} \);
2. there is a path in \( G_D^\Pi \) (for \( d \) being the density of \( \Pi \) and \( D \)) of weight \( \text{dist}(\varrho_i, \varrho_{i+1}) \) from \( v^\alpha_i \) to \( v^\alpha_{i+1} \), where \( k \) and \( \ell \) are, respectively, the \( (\Pi, D) \)-types of \( \varrho_i \) and \( \varrho_{i+1} \).

The graph \( G_D^\Pi \) can be constructed in \( L \), while checking existence of an appropriate path is feasible in NL, so existence of a derivation of \( \alpha_{i+1} \land \varrho_{i+1} \) from \( \alpha_i \land \varrho_i \) is in NL.

\[ \square \]

5 Discussion and Future Work

We have focused on fragments which mention only the past temporal operators \( \diamond \) and \( \boxdot \), however, analogous results immediately follow for the future temporal operators \( \diamondsuit \) and \( \blacklozenge \).

**Corollary 9.** Consistency checking is \( \text{TC}^0 \)-complete, \( \text{P} \)-hard, and \( \text{NL} \)-complete for DatalogMTL core DatalogMTL, and DatalogMTL core respectively, and the lower bounds hold already for the propositional fragments.

Low complexity fragments have also been identified for propositional DatalogMTL under the pointwise semantics [Kikot et al., 2018; Ryzhikov et al., 2019]. Results for pointwise and continuous semantics are, however, different: reasoning in DatalogMTL is \( \text{CoNP} \)-complete under pointwise semantics, but \( \text{PSPACE} \)-complete for continuous semantics.

We see many possibilities for future work. First, it is unclear if the bound in Theorem 7 is tight, as well as whether the upper bound applies to DatalogMTL. Second, it would be interesting to study also combined complexity. Third, we want to consider DatalogMTL over integer timelines, which may lead to lower complexity and which have been already considered for MTL [Gutiérrez-Basulto et al., 2016]. Finally, our complexity bounds do not directly provide practical reasoning algorithms, which we would like to construct.

Acknowledgments

This work was supported by the AIDA project (Alan Turing Institute), the SIRIUS Centre for Scalable Data Access (Research Council of Norway), Samsung Research UK, Siemens AG, and the EPSRC projects ANALOG (EP/P025943/1), OA-SIS (EP/S032347/1) and UK FIRES (EP/S019111/1).
References


