

# 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 TC<sup>0</sup>-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 NL-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:

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

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

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 AC<sup>0</sup> 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<sub>core</sub>, where each rule with the head different from  $\perp$  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 AC<sup>0</sup> data complexity [Calì *et al.*, 2012]. In Section 3 we show that this does not apply to our setting, as DatalogMTL<sub>core</sub> remains PSPACE-hard in data complexity, even if the *since* operator  $\mathcal{S}$  is the only temporal operator allowed in rules. Operator  $\mathcal{S}$  fully expresses the past diamond operator  $\Diamond$  (but not vice-versa), and hence the rule (1) can be written using only  $\mathcal{S}$  as a temporal operator; in contrast,  $\mathcal{S}$  does allow us to express  $\Box$ . This prompts us to investigate both DatalogMTL<sub>core</sub> <sup>$\Diamond$</sup> —the core fragment where  $\Diamond$  is the only temporal operator allowed—and the fragment DatalogMTL<sub>core</sub> <sup>$\Box$</sup> . We show that reasoning in DatalogMTL<sub>core</sub> <sup>$\Diamond$</sup>  is TC<sup>0</sup>-complete, and hence both tractable and parallelisable; however, these favourable computational properties do not transfer to DatalogMTL<sub>core</sub> <sup>$\Box$</sup> , for which reasoning is P-hard.

In Section 4 we consider DatalogMTL<sub>lin</sub>—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

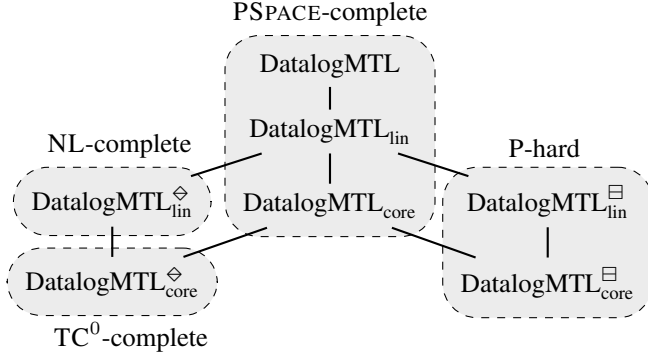


Figure 1: Hasse diagram of DatalogMTL-fragments (line segments link languages and their syntactical fragments).

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 IDB predicate Cool:

$$\text{Cool}(x) \leftarrow \diamond_{[0,2]} \text{Cool}(x) \wedge \text{Temp}(x, \text{moderate}). \quad (3)$$

DatalogMTL<sub>lin</sub> extends DatalogMTL<sub>core</sub>, and hence all lower bounds proved in Section 4 transfer. To regain favourable computational properties, we focus on DatalogMTL<sub>lin</sub><sup>◇</sup>: the linear fragment where ◇ 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  $\text{TC}^0 \subseteq \text{NL} \subseteq \text{P} \subseteq \text{PSPACE}$  and  $\text{NL} \neq \text{PSPACE}$ ).

## 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 Wałęga *et al.* (2019b), which we exploit in our technical results.

**Intervals.** We consider intervals over rationals  $\mathbb{Q}$ . An interval is denoted by  $\langle x, y \rangle$ , where the left bracket  $\langle$  is  $[$  or  $($ , the right bracket  $\rangle$  is  $]$  or  $)$ , and  $x, y \in \mathbb{Q} \cup \{-\infty, +\infty\}$ . Hence,

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

An interval  $\langle x, y \rangle$  is *positive* if  $x \geq 0$ ; it is *bounded* if  $x, y \in \mathbb{Q}$ ; 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  $\mathbb{Q}$  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  $\varrho$  a positive interval:

$$A ::= P(\tau) \mid \top \mid \perp \mid \diamond_{\varrho} A \mid \boxplus_{\varrho} A \mid \boxminus_{\varrho} A \mid \boxplus_{\varrho} A \mid AS_{\varrho} A \mid AU_{\varrho} A.$$

A literal is *EDB* if it mentions only EDB predicates and *IDB* otherwise. A (DatalogMTL) *rule* is an expression of the form

$$B \leftarrow A_1 \wedge \cdots \wedge A_n, \quad \text{for } n \geq 0, \quad (4)$$

where each  $A_i$  is a literal and  $B$  is a literal not mentioning the operators  $\diamond, \boxplus, S$ , and  $U$ . 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*  $\text{gcd}(\Pi)$  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  $\perp \leftarrow A_1 \wedge A_2$ . 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  $\perp \leftarrow A_1 \wedge A_2$ . We denote the core and linear fragments by DatalogMTL<sub>core</sub> and DatalogMTL<sub>lin</sub>, respectively. For  $X \in \{\diamond, \boxminus\}$ , and  $Y \in \{\text{core}, \text{lin}\}$ , we denote by DatalogMTL<sub>Y</sub><sup>X</sup> the fragment of DatalogMTL<sub>Y</sub> 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  $\perp$ . 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  $\alpha @ \varrho$ , where  $\alpha$  is a ground atom and  $\varrho$  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  $t \in \mathbb{Q}$ , 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  $r, s, t \in \mathbb{Q}$ . An interpretation  $\mathfrak{M}$  is a *model* of a rule of form (4) if, for each grounding assignment  $\nu$  and each  $t \in \mathbb{Q}$ , we have  $\mathfrak{M}, t \models B\nu$  whenever  $\mathfrak{M}, t \models A_i\nu$  for each  $i \in \{1, \dots, n\}$ . 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  $\alpha @ \varrho$ , written  $\mathfrak{M} \models \alpha @ \varrho$ , if  $\mathfrak{M}, t \models \alpha$  for all time points  $t \in \varrho$ , and it is a *model* of dataset  $\mathcal{D}$ , written  $\mathfrak{M} \models \mathcal{D}$ , if it is a model of all facts in  $\mathcal{D}$ . Program  $\Pi$  and dataset  $\mathcal{D}$  are *consistent* if there is a model of both  $\Pi$  and  $\mathcal{D}$ . Program  $\Pi$  and dataset  $\mathcal{D}$  *entail* fact  $\alpha @ \varrho$ , written  $(\Pi, \mathcal{D}) \models \alpha @ \varrho$ , if  $\mathfrak{M} \models \alpha @ \varrho$  for each model  $\mathfrak{M}$  of  $\Pi$  and  $\mathcal{D}$ . For brevity, we write  $\mathcal{D} \models \alpha @ \varrho$  instead of  $(\emptyset, \mathcal{D}) \models \alpha @ \varrho$ .

**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  $\alpha @ \{t\}$ , because such entailment and inconsistency checking problems are inter-reducible in all the fragments we consider. On the one hand,  $(\Pi, \mathcal{D}) \models \alpha @ \{t\}$  if and only if  $\Pi \cup \{\perp \leftarrow \alpha \wedge P\}$  and  $\mathcal{D} \cup \{P @ \{t\}\}$  are inconsistent, with



transition  $(q, a) \mapsto (q', a', L)$  is encoded by the rules

$$P_{a'} \leftarrow P_{aq}, \quad Q_{q'} \leftarrow Q_{aq}^L, \quad Q_R \leftarrow Q_{aq}, \quad Q_R \leftarrow Q_{aq}^R.$$

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  $\perp \leftarrow Q_q$  for each accepting state  $q$ , which ensures inconsistency of  $\Pi_M$  and  $\mathcal{D}_w$  when  $M$  accepts  $w$ .  $\square$

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

We start by defining a normal form. A DatalogMTL $_{\text{core}}^\diamond$  program is *normal* if each of its rules has one of the following forms, where  $\alpha$ ,  $\alpha_1$ ,  $\alpha_2$ , and  $\beta$  are atoms, and  $\varrho$  is a non-empty bounded positive interval:

$$\beta \leftarrow \diamond_{\varrho} \alpha, \quad \beta \leftarrow \top, \quad \perp \leftarrow \alpha_1 \wedge \alpha_2.$$

Each DatalogMTL $_{\text{core}}^\diamond$  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  $P(\tau) \leftarrow \diamond_{[t, \infty)} P'(\tau')$  with unbounded intervals are replaced with rules  $Q(\tau) \leftarrow \diamond_{\{t\}} P'(\tau')$ ,  $Q(\tau) \leftarrow \diamond_{[0, 1]} Q(\tau)$ , and  $P(\tau) \leftarrow \diamond_{\{0\}} Q(\tau)$ , for  $Q$  a fresh predicate of the same arity as  $P$ . Hence, in this section, we concentrate on normal DatalogMTL $_{\text{core}}^\diamond$  programs.

Next, we provide a characterisation of fact entailment for ground normal DatalogMTL $_{\text{core}}^\diamond$  programs in terms of existence of a specific length path in a graph defined below.

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

The graph  $G_\Pi^d$  is defined so that, if  $d$  is the density of  $\Pi$  and a dataset  $\mathcal{D}$ , then every derivation from  $\Pi$  and  $\mathcal{D}$  (i.e., a sequence of punctual 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  $(v_k^\alpha, v_\ell^\beta)$  of weight  $\text{dist}(\varrho_1, \varrho_2)$ , where  $\varrho_1 \ni t_1$  and  $\varrho_2 \ni t_2$  are  $(\Pi, \mathcal{D})$ -intervals of  $(\Pi, \mathcal{D})$ -types  $k$  and  $\ell$ , respectively. Indeed, the following result holds.

**Lemma 4.** Let  $\Pi$  be a ground and positive normal DatalogMTL $_{\text{core}}^\diamond$  program,  $\mathcal{D}$  a dataset, and  $d$  the density of  $\Pi$  and  $\mathcal{D}$ . Then, for each fact  $\beta @ \varrho$  with a  $(\Pi, \mathcal{D})$ -interval  $\varrho$ , the following are equivalent:

1.  $(\Pi, \mathcal{D}) \models \beta @ \varrho$ ;
2. there exist a fact  $\alpha @ \varrho'$  with  $\varrho'$  a  $(\Pi, \mathcal{D})$ -interval such that  $\mathcal{D} \models \alpha @ \varrho'$  (or  $\Pi$  contains a rule  $\alpha \leftarrow \top$ ) and a path in  $G_\Pi^d$  of weight  $\text{dist}(\varrho', \varrho)$  from  $v_k^\alpha$  to  $v_\ell^\beta$ , where  $k$  and  $\ell$  are the  $(\Pi, \mathcal{D})$ -types of  $\varrho'$  and  $\varrho$ , 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  $\text{TC}^0$  upper bound.

**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  $i$ .

*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  $\mathcal{A}$  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  $\epsilon$ -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  $\mathcal{A}$ . After eliminating  $\epsilon$ -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  $\mathcal{A}'$ , there exist  $a, b \in \mathbb{N}$  such that there is an accepting run of  $\mathcal{A}'$  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).  $\square$

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

To check consistency of a ground normal DatalogMTL $_{\text{core}}^\diamond$  program  $\Pi$  and a dataset  $\mathcal{D}$ , we need to consider rules of the form  $\perp \leftarrow \alpha_1 \wedge \alpha_2$ , 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  $\mathcal{D}$  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 $_{core}^\diamond$  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  $\{\perp \leftarrow A \wedge B\}$ , for propositions  $A$  and  $B$ , and the dataset  $\{A @ \{a\}, B @ \{\frac{c}{b}\}\}$  are inconsistent.

For the upper bound, we first consider a ground normal DatalogMTL $_{core}^\diamond$  program  $\Pi$ , with  $\Pi^+$  its positive subset. Since rules in  $\Pi$  with  $\perp$  do not contain temporal operators, the  $(\Pi, \mathcal{D})$ - and  $(\Pi^+, \mathcal{D})$ -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  $\mathcal{D}$  are inconsistent if and only if there is a rule  $\perp \leftarrow \beta_1 \wedge \beta_2$  in  $\Pi$  and a subset  $\mathcal{D}'$  of  $\mathcal{D}$  with at most two facts such that  $(\Pi^+, \mathcal{D}') \models \beta_i @ \varrho$  for both  $i$  and some  $(\Pi^+, \mathcal{D}')$ -interval  $\varrho$ . Consider such  $\mathcal{D}' = \{\alpha_1 @ \varrho_1, \alpha_2 @ \varrho_2\}$ . Moreover, for both  $i$ , let  $\varrho_i^L$  and  $\varrho_i^R$  be the left- and the right-most  $(\Pi^+, \mathcal{D}')$ -intervals in  $\varrho_i$ , respectively, let  $K_i$  be the sets of the  $(\Pi^+, \mathcal{D}')$ -types of all  $(\Pi^+, \mathcal{D}')$ -intervals in  $\varrho_i$ , and let  $d$  be the density of  $\Pi^+$  and  $\mathcal{D}'$ . Then, assuming for simplicity that  $\Pi$  does not have rules  $\alpha \leftarrow \top$ , we use Lemma 4 to reduce existence of  $\varrho$  as above to existence of two paths in  $G_{\Pi^+}^d$ : one from  $v_{k_1}^{\alpha_1}$  to  $v_{\ell}^{\beta_1}$  with weight  $n_1$  and another from  $v_{k_2}^{\alpha_2}$  to  $v_{\ell}^{\beta_2}$  with weight  $n_2$ , where  $k_i \in K_i$  for both  $i$ ,  $\ell$  is a  $(\Pi^+, \mathcal{D}')$ -type, and

$$\text{dist}(\varrho_2^R, \varrho_1^L) \leq n_1 - n_2 \leq \text{dist}(\varrho_2^L, \varrho_1^R). \quad (5)$$

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  $v_{k_i}^{\alpha_i}$  to  $v_{\ell}^{\beta_i}$  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

$$\text{dist}(\varrho_2^R, \varrho_1^L) \leq (a_1 + i_1 \cdot b_1) - (a_2 + i_2 \cdot b_2) \leq \text{dist}(\varrho_2^L, \varrho_1^R),$$

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

$$\left[ \frac{\text{dist}(\varrho_2^R, \varrho_1^L) - a_1 + a_2}{\gcd(b_1, b_2)}, \frac{\text{dist}(\varrho_2^L, \varrho_1^R) - a_1 + a_2}{\gcd(b_1, b_2)} \right].$$

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  $\mathcal{D}$ .

Finally, let  $\Pi$  be a non-ground normal DatalogMTL $_{core}^\diamond$  program. Note that although the grounding  $\Pi_{\mathcal{D}}$  of  $\Pi$  with constants from  $\mathcal{D}$  can be computed in  $AC^0$ , the graph  $G_{\Pi_{\mathcal{D}}}^d$  depends not only on  $\Pi$  and  $d$  but also on  $\mathcal{D}$ ; this would prevent us from exploiting the  $TC^0$  procedure for ground programs. However, if  $\mathcal{D}'$  and  $\mathcal{D}''$  are the same modulo renaming of constants not occurring in  $\Pi$ , then  $G_{\Pi_{\mathcal{D}'}}^k$  and  $G_{\Pi_{\mathcal{D}''}}^k$  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  $\mathcal{D}$ . 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.  $\square$

Now, we show P-hardness for consistency checking in DatalogMTL $_{core}^\square$ , that holds even for propositional programs.

**Theorem 7.** *Consistency checking in propositional DatalogMTL $_{core}^\square$  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  $(V, E, S, v_t)$ , where  $V = \{v_1, \dots, v_n\}$  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  $v_t \in V$  is a (target) node. The answer to  $G$  is affirmative if  $v_t$  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  $(v_i, v_j, v_k) \in E$ .

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

$$\begin{aligned} P' &\leftarrow \Box_{\{2\}} P, & P'' &\leftarrow \Box_{(0,1)} P', & P &\leftarrow \Box_{\{2\}} P'', \\ P &\leftarrow \Box_{\{4\}} P, & \perp &\leftarrow P \wedge Q. \end{aligned}$$

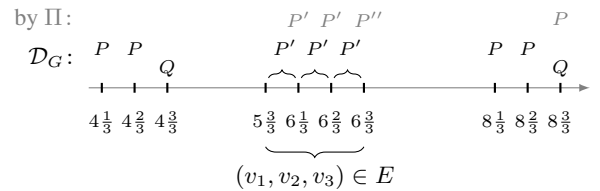
Consider now a PSA instance  $G$  as above, and let  $e_1, \dots, e_m$  be an enumeration of  $E$  preserving the ‘application order’—that is, such that for every edge  $e_\ell$  with node  $v_k$  as the third argument,  $v_k$  is not the first or second argument of any  $e_h$  with  $h < \ell$ . Let  $\mathcal{D}_G$  be a dataset containing the following facts, for every  $\ell$  such that  $1 \leq \ell \leq m+1$ :

$$Q @ \{4\ell + t/n\} \quad \text{and} \quad P @ \{4\ell + s/n\}, \quad \text{for every } v_s \in S,$$

and the following facts, for every  $e_\ell = (v_i, v_j, v_k)$ :

$$\begin{aligned} P' @ (4\ell + 1 + k/n, 4\ell + 2 + i/n), \\ P' @ (4\ell + 2 + i/n, 4\ell + 2 + j/n), \\ P' @ (4\ell + 2 + j/n, 4\ell + 2 + k/n). \end{aligned}$$

For instance,  $\mathcal{D}_G$ , 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.



It can then be easily checked that  $v_t$  is accessible if and only if  $\Pi$  and  $\mathcal{D}_G$  are inconsistent.  $\square$

## 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 $_{lin}^\diamond$  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<sub>lin</sub><sup>◇</sup> programs.

For the upper bound, we will concentrate on programs in a normal form, extending the one for the core case; a DatalogMTL<sub>lin</sub><sup>◇</sup> 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:

$$\beta \leftarrow \Diamond_{\varrho} \alpha \wedge \phi, \quad \beta \leftarrow \top, \quad \perp \leftarrow \alpha_1 \wedge \alpha_2.$$

Each DatalogMTL<sub>lin</sub><sup>◇</sup> 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  $\perp$  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<sub>lin</sub><sup>◇</sup> is NL-complete in data complexity. The lower bound holds already for propositional DatalogMTL<sub>lin</sub><sup>◇</sup>.*

*Proof sketch.* The lower bound is by reduction of consistency checking for propositional normal DatalogMTL<sub>core</sub><sup>◇</sup> 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<sub>core</sub><sup>◇</sup> program  $\Pi$ , we introduce a fresh IDB proposition  $Q_A$  for each literal  $A$  of the form  $\Diamond_{\varrho} P$  in  $\Pi$ , and a fresh EDB proposition  $Q$ . We then construct a propositional DatalogMTL<sub>lin</sub><sup>◇</sup> program  $\Pi'$  by replacing each literal  $A$  as above in  $\Pi$  by  $Q_A$  and adding rules  $Q_A \leftarrow A \wedge Q$ . For each (punctual) dataset  $\mathcal{D}$ ,  $\Pi$  and  $\mathcal{D}$  are consistent under pointwise semantics if and only if so are  $\Pi'$  and  $\mathcal{D} \cup \{Q @ \varrho \mid \varrho \text{ occurs in } \mathcal{D}\}$  under continuous semantics.

For the upper bound, consider a normal DatalogMTL<sub>lin</sub><sup>◇</sup> program  $\Pi$  and a dataset  $\mathcal{D}$ . By Lemma 1,  $\Pi$  and  $\mathcal{D}$  are inconsistent if and only if there are a grounding  $\perp \leftarrow \beta_1 \wedge \beta_2$  of a rule in  $\Pi$  and derivations of both facts  $\beta_i @ \varrho$  for some  $(\Pi, \mathcal{D})$ -interval  $\varrho$ . 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  $\mathcal{D}$ . For each  $\phi \in \Phi$ , we compute the maximal intervals in which  $\mathcal{D}$  entails  $\phi$ , which is feasible in TC<sup>0</sup>. 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  $\mathcal{D}$  entails  $\phi \in \Phi$  in some time point in a section  $\sigma$ , then  $\mathcal{D}$  entails  $\phi$  in the whole  $\sigma$ . Let  $\Phi_{\sigma}$  be the set of all  $\phi \in \Phi$  entailed in a section  $\sigma$  by  $\mathcal{D}$ . 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  $\mathcal{D}$

closest to  $\varrho$  and the number  $\text{dist}(\{t\}, \varrho)$ , which is polynomially representable in the size of  $\mathcal{D}$  (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  $\alpha_{i+1} @ \varrho_{i+1}$  in the subsequence can be derived from the preceding fact  $\alpha_i @ \varrho_i$ . If  $\varrho_i$  and  $\varrho_{i+1}$  are in different sections, then we check, in L, whether there is a 1-step derivation of  $\alpha_{i+1} @ \varrho_{i+1}$  from  $\alpha_i @ \varrho_i$ . Otherwise—that is, when  $\varrho_i$  and  $\varrho_{i+1}$  are in the same section  $\sigma$ —the derivation of  $\alpha_{i+1} @ \varrho_{i+1}$  from  $\alpha_i @ \varrho_i$  can be of exponential length. To deal with this case, 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<sub>core</sub><sup>◇</sup> 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  $\mathcal{D}$  that starts with  $\alpha_i @ \varrho_i$  and ends with  $\alpha_{i+1} @ \varrho_{i+1}$ ;
2. there is a path in  $G_{\Pi_{\sigma}}^d$  (for  $d$  being the density of  $\Pi$  and  $\mathcal{D}$ ) of weight  $\text{dist}(\varrho_i, \varrho_{i+1})$  from  $v_{\ell}^{\alpha_i}$  to  $v_{\ell}^{\alpha_{i+1}}$ , where  $k$  and  $\ell$  are, respectively, the  $(\Pi, \mathcal{D})$ -types of  $\varrho_i$  and  $\varrho_{i+1}$ .

The graph  $G_{\Pi_{\sigma}}^d$  can be constructed in L, while checking existence of an appropriate path is feasible in NL, so existence of a derivation of  $\alpha_{i+1} @ \varrho_{i+1}$  from  $\alpha_i @ \varrho_i$  is in NL.  $\square$

## 5 Discussion and Future Work

We have focused on fragments which mention only the past temporal operators  $\Diamond$  and  $\Box$ , however, analogous results immediately follow for the future temporal operators  $\Diamond$  and  $\Box$ .

**Corollary 9.** *Consistency checking is TC<sup>0</sup>-complete, P-hard, and NL-complete for DatalogMTL<sub>core</sub><sup>◇</sup>, DatalogMTL<sub>core</sub><sup>□</sup>, and DatalogMTL<sub>lin</sub><sup>◇</sup>, 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 CONP-complete under pointwise semantics, but 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<sub>lin</sub><sup>□</sup>. 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

- [Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [Artale *et al.*, 2017] Alessandro Artale, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyashev. Ontology-mediated query answering over temporal data: A survey (invited talk). In *TIME*, 2017.
- [Brandt *et al.*, 2017] Sebastian Brandt, Roman Kontchakov, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyashev. Ontology-based data access with a Horn fragment of metric temporal logic. In *AAAI*, pages 1070–1076, 2017.
- [Brandt *et al.*, 2018] Sebastian Brandt, Elem Güzel Kalaycı, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyashev. Querying log data with metric temporal logic. *J. Artif. Intell. Res.*, 62:829–877, 2018.
- [Calì *et al.*, 2012] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Semant.*, 14:57–83, 2012.
- [Chrobak, 1986] Marek Chrobak. Finite automata and unary languages. *Theor. Comput. Sci.*, 47:149–158, 1986.
- [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.
- [Gutiérrez-Basulto *et al.*, 2016] Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Ana Ozaki. On metric temporal description logics. In *ECAI*, pages 837–845, 2016.
- [Hesse, 2001] William Hesse. Division is in uniform  $TC^0$ . In *ICALP*, pages 104–114. Springer, 2001.
- [Hunter *et al.*, 2013] Paul Hunter, Joël Ouaknine, and James Worrell. Expressive completeness for metric temporal logic. In *LICS*, pages 349–357, 2013.
- [Kikot *et al.*, 2018] Stanislav Kikot, Vladislav Ryzhikov, Przemysław Andrzej Wałęga, and Michael Zakharyashev. On the data complexity of ontology-mediated queries with MTL operators over timed words. In *DL*, 2018.
- [Motik *et al.*, 2012] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web ontology language profiles (2nd edition), 2012. W3C Recommendation.
- [Ouaknine and Worrell, 2008] Joël Ouaknine and James Worrell. Some recent results in metric temporal logic. In *FORMATS*, pages 1–13, 2008.
- [Ryzhikov *et al.*, 2019] Vladislav Ryzhikov, Przemysław Andrzej Wałęga, and Michael Zakharyashev. Data complexity and rewritability of ontology-mediated queries in metric temporal logic under the event-based semantics. In *IJCAI*, pages 1851–1857, 2019.
- [Wałęga *et al.*, 2019a] Przemysław Andrzej Wałęga, Bernardo Cuenca Grau, and Mark Kaminski. Reasoning over streaming data in metric temporal Datalog. In *AAAI*, pages 1941–1948, 2019.
- [Wałęga *et al.*, 2019b] Przemysław Andrzej Wałęga, Bernardo Cuenca Grau, Mark Kaminski, and Egor V. Kostylev. DatalogMTL: Computational complexity and expressive power. In *IJCAI*, pages 1886–1892, 2019.
- [Xiao *et al.*, 2018] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyashev. Ontology-based data access: A survey. In *IJCAI*, pages 5511–5519, 2018.