

DatalogMTL over Integer Timeline

Przemysław A. Wałęga, Bernardo Cuenca Grau, Mark Kaminski, 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 DatalogMTL—an extension of Datalog with metric temporal operators—under integer semantics, where the temporal domain of both interpretations and temporal operators consists of integer time points only. This is in contrast to the standard semantics, which is defined over the rational timeline. DatalogMTL under integer semantics is an interesting language for knowledge representation: on the one hand, one can often assume the integer timeline in applications; on the other hand, DatalogMTL under integer semantics strictly captures prominent temporal extensions of Datalog such as Datalog_{IS}. We show that the integer semantics leads to more favourable computational properties than the rational semantics. We first show that reasoning over integers is at most as hard as reasoning over rationals for DatalogMTL and its natural fragments. Then, we investigate syntactic fragments of DatalogMTL where adopting the integer semantics makes reasoning significantly easier. In particular, we show that complexity drops from P-hard to NC¹-complete for the propositional fragment (where all object variables are grounded), and from TC⁰-hard to ACC⁰ for the linear fragment where the past diamond operator is the only metric operator allowed in rule bodies. Thus, reasoning in such fragments is both tractable and highly parallelisable, which suggests their appropriateness for data-intensive applications.

1 Introduction

DatalogMTL (Brandt et al. 2018; Brandt et al. 2017) is a recently introduced temporal extension of Datalog, where atoms in rules can mention operators from metric temporal logic MTL (Koymans 1990) interpreted over the rational timeline. For example, the ground expression $\Diamond_{[1,10]}Temp(c, high)$ holds at time t (e.g., expressed in seconds) if sensor c reported a high temperature sometime within the interval $[t - 10, t - 1]$. Similarly, $\Box_{[1,10]}Temp(c, high)$ holds at t if the sensor reported high temperature continuously in the aforementioned interval. Then, the rule (1) triggers an alarm for sensor x at time t if the sensor reported high temperature at least once every ten seconds within the previous minute:

$$Alarm(x) \leftarrow \Box_{[1,60]} \Diamond_{[1,10]} Temp(x, high). \quad (1)$$

A dataset consists of facts involving intervals, such as $Temp(c, high)@[15, 21]$, stating that $Temp(c, high)$ holds continuously in the interval $[15, 21]$.

DatalogMTL provides a powerful language which is suitable for ontology-based data access OBDA (Brandt et al. 2018), stream reasoning (Wałęga, Cuenca Grau, and Kaminski 2019), and temporal logic programming (Brzoska 1998). Thus, the computational properties of DatalogMTL and its fragments have recently received attention in the literature (Wałęga et al. 2019; Ryzhikov, Wałęga, and Zakharyashev 2019; Wałęga et al. 2019).

In the standard semantics, DatalogMTL is interpreted over the rational timeline. Having a dense time domain provides a great deal of modelling flexibility, which can be beneficial in applications (Brandt et al. 2018). In many cases, however, it suffices to consider only discretely ordered time points, for instance, when measuring devices produce readings with a given frequency or signals are processed according to a fixed global clock. Then, it may be worth considering the integer timeline—a possibility that has already been explored for MTL (Ouaknine and Worrell 2008) and exploited in applications such as the control of unmanned aircrafts (Doherty, Kvarnström, and Heintz 2009), as well as for metric extensions of description logics (Gutiérrez-Basulto, Jung, and Ozaki 2016; Baader et al. 2017). Adopting integer semantics changes the meaning of temporal expressions such as $\Diamond_{[1,10]}Temp(c, high)$ which now holds at t if a high temperature was reported in one of the last 10 integer time points rather than in one of the (infinitely many) rational time points in the interval $[t - 10, t - 1]$.

In this paper, we study the computational complexity of reasoning in DatalogMTL over the integer timeline, as defined in Section 2. **!** In such a setting, the semantics of DatalogMTL is closely related to the semantics of linear temporal logic (Artale et al. 2017; Gutierrez Basulto, Jung, and Kontchakov 2016), however, facts and operators in DatalogMTL are much more succinct due to the binary encoding of numbers. We will consider DatalogMTL fragments along two dimensions. On the one hand, we study restrictions on the allowed operators and consider fragments allowing only for \Box or only for \Diamond as metric operators; on the other hand, we study restrictions on the structure of rules and consider both *linear* fragments, where rule bodies contain at most one intensional (IDB) conjunct, and *core* fragments, where additionally rules without \perp in the head contain a single conjunct in the body (note that rule (1) is both linear and core). In the setting of the integer timeline,

P: a new
fragment starts
here (with a
new citations
added)

DatalogMTL remains a powerful KR language, which can capture well-known temporal formalisms such as Datalog with a unary successor operator Datalog_{1S} (Chomicki and Imieliński 1988) and some Horn fragments of Linear Temporal Logic (Artale et al. 2017).

Our technical results are as follows. In Section 3, we show that reasoning under integer semantics in DatalogMTL reduces to reasoning over the rationals, and moreover the reduction preserves data complexity. As a result, data complexity upper bounds over rationals transfer to the integer case and thus reasoning in DatalogMTL remains in PSPACE under integer semantics. The matching lower bound is inherited from known results on Datalog_{1S} (Chomicki and Imieliński 1988), but we also show PSPACE-hardness for the linear fragment allowing only for \Box among metric operators.

In Section 4 we consider the propositional fragment of DatalogMTL, where rules contain no variables. There are two different factors contributing to the size of a dataset in DatalogMTL, namely the object constants and timestamps occurring in facts. The propositional fragment is relevant because it showcases the effect of the latter source of complexity, which is exclusive to temporal data. In practice, this can be the dominating source of complexity; for example in stream reasoning applications where constants (e.g., sensors) can often be treated as fixed, whereas the number of timestamps occurring in facts can be huge (or even unbounded). It was shown in (Brandt et al. 2018) that reasoning in the propositional fragment is P-hard in data complexity over rationals, where the best known upper bound is PSPACE (Wałęga et al. 2019). We show that complexity drops significantly under integer semantics, and reasoning becomes NC^1 -complete and hence both tractable and parallelisable. Furthermore, the NC^1 lower bound holds already for the linear fragment with only \Diamond and for the core fragment with only \Box .

In Section 5 we focus on the core fragment with only \Diamond , which is not covered by the NC^1 lower bounds in Section 4. This fragment is interesting in its own right as it captures the core forward-propagating fragment of Datalog_{1S} . It is straightforward that, over rationals, reasoning in this fragment is TC^0 -hard already in the propositional case as it can simulate integer multiplication. In contrast, we show that under integer semantics data complexity drops to ACC^0 even for the non-propositional case, and we also show that the complexity is outside AC^0 even in the propositional case.

Our results cover a wide range of relevant fragments of DatalogMTL for which reasoning over the integer timeline becomes easier than over rationals. In these cases, we can show very low data complexity which suggests the applicability of these fragments in data-extensive applications.

2 Preliminaries

Temporal Domains. We consider DatalogMTL and its fragments over two *temporal domains*: rationals \mathbb{Q} and integers \mathbb{Z} . Hence, all our relevant notions are parametrised by a temporal domain $\mathbb{T} \in \{\mathbb{Q}, \mathbb{Z}\}$ —and so we have \mathbb{T} -intervals, \mathbb{T} -programs, \mathbb{T} -interpretations, \mathbb{T} -consistency checking, \mathbb{T} -entailment, etc.; but, we will usually not mention the domain explicitly and omit \mathbb{T} -prefix when it is clear from the

context. We assume that each integer is represented in binary and each rational as a pair of an integer numerator and a positive integer denominator (both in binary).

Intervals. We consider \mathbb{T} -intervals $\langle t_1, t_2 \rangle$, where the left bracket \langle is either $[$ or $($, the right bracket \rangle is either $]$ or $)$, and $t_1, t_2 \in \mathbb{T} \cup \{-\infty, \infty\}$. Hence,

$$\langle t_1, t_2 \rangle = \{ t \in \mathbb{T} \mid t_1 \leq t \leq t_2, \text{ if } \langle \text{ is } (\text{ then } t \neq t_1, \\ \text{and if } \rangle \text{ is }) \text{ then } t \neq t_2 \}.$$

We use \langle and \rangle for unspecified brackets. An interval $\langle t_1, t_2 \rangle$ is *positive* if $t_1 \geq 0$; it is *bounded* if $t_1, t_2 \in \mathbb{T}$; and it is *punctual* if it is of the form $[t, t]$ —then, we write $\{t\}$.

Syntax. We assume a function-free first-order vocabulary. An *atom* is an expression of the form $P(\tau)$ with P a predicate and τ a tuple of constants and variables of matching arity. Each predicate is either *extensional* (EDB) or *intensional* (IDB). A *literal* A is an expression given by the following grammar, with α an atom and ϱ a positive interval:

$$A ::= \alpha \mid \top \mid \perp \mid \Diamond_{\varrho} A \mid \Box_{\varrho} A \mid \Box_{\varrho} A \mid \Box_{\varrho} A \mid AS_{\varrho} A \mid AU_{\varrho} A.$$

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

$$B \leftarrow A_1 \wedge \dots \wedge A_n, \quad \text{for } n \geq 0, \quad (2)$$

where each A_i are literals and B is an IDB literal not mentioning the operators \Diamond , \Box , S , and U . The conjunction $A_1 \wedge \dots \wedge A_n$ constitutes the rule *body*, whereas literal B is the *head*. A rule is *safe* if each head variable occurs also in the body. A DatalogMTL *program* is a finite set of safe rules. A program is *core* if each of its rules is of the form $\perp \leftarrow A_1 \wedge A_2$ or $B \leftarrow A$. It is *linear* if each rule is either of the form (2) with at most one A_i being IDB, or of the form $\perp \leftarrow A_1 \wedge A_2$. For $X \in \{\Diamond, \Box\}$ and $Y \in \{\text{core}, \text{lin}\}$, we denote by DatalogMTL_Y^X the fragment where programs are core if Y is core or linear if Y is lin, and X is the only temporal operator allowed in literals. A program is *propositional* if all its predicates are nullary (i.e., propositions). An expression (e.g., an atom or a literal) is *ground* if it mentions no variables. A *fact* is an expression of the form $\alpha @ \varrho$, with α a ground atom and ϱ 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 α and time point $t \in \mathbb{T}$, whether α is true at t , in which case we write $\mathfrak{M}, t \models_{\mathbb{T}} \alpha$. This notion extends to ground literals as in Table 1. Interpretation \mathfrak{M} is a *model* of a rule (2) if, for each assignment ν of constants to variables grounding the rule and each $t \in \mathbb{T}$, we have $\mathfrak{M}, t \models_{\mathbb{T}} B\nu$ whenever $\mathfrak{M}, t \models_{\mathbb{T}} A_i\nu$ for each $i \in \{1, \dots, n\}$. It is a *model* of a program Π , written $\mathfrak{M} \models_{\mathbb{T}} \Pi$, if it is a model of all rules in Π . It is a *model* of a fact $\alpha @ \varrho$, written $\mathfrak{M} \models_{\mathbb{T}} \alpha @ \varrho$, if $\mathfrak{M}, t \models_{\mathbb{T}} \alpha$ for all $t \in \mathbb{T}$ within ϱ . Finally, it is a *model* of a dataset \mathcal{D} , if it is a model of all facts in \mathcal{D} . A program Π and a dataset \mathcal{D} are *consistent* if they admit a common model; they *entail* a fact $\alpha @ \varrho$, written $(\Pi, \mathcal{D}) \models_{\mathbb{T}} \alpha @ \varrho$, if $\mathfrak{M} \models_{\mathbb{T}} \alpha @ \varrho$ for each common model \mathfrak{M} . We write $\mathcal{D} \models_{\mathbb{T}} \alpha @ \varrho$ instead of $(\emptyset, \mathcal{D}) \models_{\mathbb{T}} \alpha @ \varrho$.

P: a new
fragment starts
here

$\mathfrak{M}, t \models_{\mathbb{T}} \top$	for each t
$\mathfrak{M}, t \not\models_{\mathbb{T}} \perp$	for each t
$\mathfrak{M}, t \models_{\mathbb{T}} \Diamond_{\varrho} A$	if $\exists t'$ such that $t - t' \in \varrho$ and $\mathfrak{M}, t' \models_{\mathbb{T}} A$
$\mathfrak{M}, t \models_{\mathbb{T}} \Box_{\varrho} A$	if $\exists t'$ such that $t' - t \in \varrho$ and $\mathfrak{M}, t' \models_{\mathbb{T}} A$
$\mathfrak{M}, t \models_{\mathbb{T}} \Box_{\varrho} A$	if $\forall t'$ if $t - t' \in \varrho$ then $\mathfrak{M}, t' \models_{\mathbb{T}} A$
$\mathfrak{M}, t \models_{\mathbb{T}} \Box_{\varrho} A$	if $\forall t'$ if $t' - t \in \varrho$ then $\mathfrak{M}, t' \models_{\mathbb{T}} A$
$\mathfrak{M}, t \models_{\mathbb{T}} AS_{\varrho} A'$	if $\exists t'$ such that $t - t' \in \varrho$, $\mathfrak{M}, t' \models_{\mathbb{T}} A'$, and $\forall t''$ if $t'' \in (t', t)$ then $\mathfrak{M}, t'' \models_{\mathbb{T}} A$
$\mathfrak{M}, t \models_{\mathbb{T}} AU_{\varrho} A'$	if $\exists t'$ such that $t' - t \in \varrho$, $\mathfrak{M}, t' \models_{\mathbb{T}} A'$, and $\forall t''$ if $t'' \in (t, t')$ then $\mathfrak{M}, t'' \models_{\mathbb{T}} A$

Table 1: Semantics of ground literals, where \mathfrak{M} is an \mathbb{T} -interpretation and $t, t', t'' \in \mathbb{T}$

Reasoning Problems. We study the *data complexity* of \mathbb{T} -consistency checking for DatalogMTL and its fragments—that is, we assume that a program is fixed while only the dataset constitutes the input. All our complexity results are also applicable to entailment of facts with punctual intervals for which entailment and inconsistency checking are interreducible for all temporal domains and fragments we consider. Indeed, $(\Pi, \mathcal{D}) \models_{\mathbb{T}} \alpha @ \{t\}$ if and only if $\Pi \cup \{\perp \leftarrow \alpha \wedge P\}$ and $\mathcal{D} \cup \{P @ \{t\}\}$ are inconsistent, for P a fresh proposition; whereas Π and \mathcal{D} are inconsistent if and only if Π and \mathcal{D} entail a fact with a fresh proposition. Moreover, by constructing reductions involving \mathcal{S} and \mathcal{U} one can show that in DatalogMTL the data complexity of entailment for facts with arbitrary intervals is the same as the complexity of inconsistency checking. However, it is not known whether the same holds for DatalogMTL fragments without \mathcal{S} and \mathcal{U} .

Complexity Classes. We assume familiarity with basic complexity classes such as P and PSPACE. Our results will also involve the following complexity classes defined via DLOGTIME-uniform families of polynomial-size Boolean circuits (Papadimitriou 2003): AC^0 , where AND and OR gates may have unbounded fan-in and circuit depth is constant; $\text{AC}^0[c_1, \dots, c_k]$, extending AC^0 by additionally allowing modulo c_i gates (of unbounded fan-in) for all $i \in \{1, \dots, k\}$; ACC^0 , which is the same as $\text{AC}^0[c_1, \dots, c_k]$ except that it allows modulo c gates for all c ; TC^0 , extending AC^0 by allowing majority gates (also of unbounded fan-in); and NC^1 , which is the same as AC^0 , except that all gates have fan-in at most 2 but the depth is logarithmic. It is known that $\text{AC}^0 \subseteq \text{AC}^0[c_1, \dots, c_k] \subseteq \text{ACC}^0 \subseteq \text{TC}^0 \subseteq \text{NC}^1 \subseteq \text{P} \subseteq \text{PSPACE}$, $\text{AC}^0 \subsetneq \text{AC}^0[c]$ if $c > 1$, $\text{AC}^0[c] \subsetneq \text{TC}^0$ if c is a power of prime, and $\text{NC}^1 \subsetneq \text{PSPACE}$.

3 Rational vs Integer Semantics

In this section we compare the integer and rational semantics of DatalogMTL. Each \mathbb{Z} -interval is clearly a \mathbb{Q} -interval, and

so every \mathbb{Z} -program is a \mathbb{Q} -program and each \mathbb{Z} -dataset is a \mathbb{Q} -dataset. The notions of \mathbb{T} -interpretation and the satisfaction relation $\models_{\mathbb{T}}$, however, critically depend on the temporal domain \mathbb{T} . In particular, a \mathbb{Z} -dataset may entail different facts over \mathbb{Z} -intervals depending on whether one considers integer or rational semantics; thus, it is not the case that a \mathbb{Z} -program and a \mathbb{Z} -dataset are \mathbb{Z} -consistent if and only if they are \mathbb{Q} -consistent. For instance, consider a \mathbb{Z} -program $\Pi = \{\perp \leftarrow \Box_{(0,1]} P\}$ and a \mathbb{Z} -dataset $\mathcal{D} = \{P @ \{0\}\}$. In every \mathbb{Z} -model \mathfrak{M} of \mathcal{D} we have $\mathfrak{M}, 1 \models_{\mathbb{Z}} \Box_{(0,1]} P$ and so Π and \mathcal{D} are \mathbb{Z} -inconsistent; on the other hand, it is not the case that in every \mathbb{Q} -model \mathfrak{M} of \mathcal{D} we have $\mathfrak{M}, 1 \models_{\mathbb{Q}} \Box_{(0,1]} P$, and it is easy to see that Π and \mathcal{D} are \mathbb{Q} -consistent. Thus, both semantics are fundamentally different and complexity results cannot be directly transferred from one to the other.

In what follows, we show that consistency checking over integers reduces to consistency checking over rationals, and the reduction preserves the data complexity.

Theorem 1. *For DatalogMTL, checking whether a \mathbb{Z} -program Π and a \mathbb{Z} -dataset \mathcal{D} are \mathbb{Z} -consistent reduces to checking whether a \mathbb{Q} -program Π' and a \mathbb{Q} -dataset \mathcal{D}' are \mathbb{Q} -consistent, where Π' depends only on Π and \mathcal{D}' is constructed from \mathcal{D} in AC^0 . The same statement holds for DatalogMTL_Y^X , for every $X \in \{\Diamond, \Box\}$ and $Y \in \{\text{core}, \text{lin}\}$.*

Proof sketch. For each literal A , we define literal A' by first replacing each interval ϱ in A with the maximal \mathbb{Z} -interval included in ϱ of the form $[t_1, t_2]$ if ϱ is bounded and $[t_1, \infty)$ if ϱ is unbounded, and then replacing each sub-literal of the form $A_1 S_{\varrho} A_2$ or $A_1 U_{\varrho} A_2$ with $\Diamond_{[0,1]} A_1 S_{\varrho} A_2$ or $\Diamond_{[0,1]} A_1 U_{\varrho} A_2$, respectively. Let now Π be a DatalogMTL \mathbb{Z} -program, and let Π' be obtained from Π by replacing each literal A with A' . Note that if Π is a DatalogMTL_Y^X \mathbb{Z} -program then so is Π' (and so Π' is a DatalogMTL_Y^X \mathbb{Q} -program).

Next, given a \mathbb{Z} -dataset \mathcal{D} we will construct a \mathbb{Q} -dataset \mathcal{D}' by modifying intervals in \mathcal{D} in two steps. In the first step we replace every left-open interval $\langle t_1, t_2 \rangle$ in \mathcal{D} such that $t_1 \in \mathbb{Z}$ with $[t_1 + 1, t_2]$; and in the second step we replace every right-closed interval $\langle t_1, t_2 \rangle$ such that $t_2 \in \mathbb{Z}$ with $\langle t_1, t_2 + 1 \rangle$. The construction is in AC^0 and every interval in \mathcal{D}' is either of the form $[t_1, t_2]$ or $(-\infty, t_2]$, where $t_1 \in \mathbb{Z}$ and $t_2 \in \mathbb{Z} \cup \{\infty\}$.

It remains to show that Π and \mathcal{D} are \mathbb{Z} -consistent if and only if Π' and \mathcal{D}' are \mathbb{Q} -consistent. First, we define a transformation f mapping each \mathbb{Z} -interpretation \mathfrak{M} to a \mathbb{Q} -interpretation $f(\mathfrak{M})$ such that $f(\mathfrak{M}), t \models_{\mathbb{Q}} \alpha$ if and only if $\mathfrak{M}, \lfloor t \rfloor \models_{\mathbb{Z}} \alpha$, for every $t \in \mathbb{Q}$ and ground atom α , where $\lfloor \cdot \rfloor$ is the floor function. We can show inductively on the structure of literals that $(\star) \mathfrak{M} \models_{\mathbb{Z}} A(\tau) @ \{t\}$ if and only if $f(\mathfrak{M}) \models_{\mathbb{Q}} A'(\tau) @ [t, t + 1]$, for every \mathbb{Z} -interpretation \mathfrak{M} , literal A in Π , tuple of constants τ , and integer t . If \mathfrak{M} is a \mathbb{Z} -model of Π and \mathcal{D} , then $f(\mathfrak{M})$ is a \mathbb{Q} -model of Π' and \mathcal{D}' and hence Π' and \mathcal{D}' are \mathbb{Q} -consistent. Conversely, if Π' and \mathcal{D}' are \mathbb{Q} -consistent, then we can show, following (Wałęga et al. 2019, Lemma 4), that there exists the unique least \mathbb{Q} -model \mathfrak{M}' of Π' and \mathcal{D}' (i.e., the \mathbb{Q} -model assigning true as little as possible), and moreover that $\mathfrak{M}' \models_{\mathbb{Q}} \alpha @ \{t\}$ implies

$\mathcal{M}' \models_{\mathbb{Q}} \alpha @ [t, t] + 1$, for every ground atom α and rational t . Thus, $f^{-1}(\mathcal{M}')$ is well-defined and (\star) ensures that $f^{-1}(\mathcal{M}')$ is a \mathbb{Z} -model of Π and \mathcal{D} . \square

Thus, data complexity upper bounds for \mathbb{Q} -consistency checking transfer to \mathbb{Z} -consistency checking. It hence follows from the results of Wałęga et al. (2019) for rationals that the data complexity of \mathbb{Z} -consistency checking in DatalogMTL is in PSPACE. The matching lower bound is obtained by a simple reduction of consistency checking in Datalog_{IS} (Chomicki and Imieliński 1988), where the successor operator is encoded as $\Box_{\{1\}}$ (or equivalently $\Diamond_{\{1\}}$). We next prove PSPACE-hardness of \mathbb{Z} -consistency already for linear programs mentioning only \Box among temporal operators. By Theorem 1, this also constitutes a (new) lower bound of \mathbb{Q} -consistency checking.

Theorem 2. *Checking \mathbb{Z} -consistency in DatalogMTL is PSPACE-complete in data complexity, and the lower bound holds already for DatalogMTL_{lin} ^{\Box} .*

Proof sketch. The upper bound follows from Theorem 1 and the results of Wałęga et al. (2019). For the lower bound, we provide a reduction of an arbitrary problem \mathcal{P} in PSPACE to \mathbb{Z} -consistency checking. Let M be a deterministic Turing machine and p a polynomial such that, for every input word w , the machine decides \mathcal{P} for w using at most $p(|w|)$ cells. We construct, in logarithmic space, a \mathbb{Z} -dataset \mathcal{D}_w encoding the initial configuration of M on w and a DatalogMTL_{lin} ^{\Box} \mathbb{Z} -program Π_M simulating the computations of M (on any input word) such that M accepts w if and only if Π_M and \mathcal{D}_w are \mathbb{Z} -inconsistent. In particular, we will encode the i -th configuration of M by a set of atoms holding at a time point $5 \cdot i$, where the gaps (of length 5) between descriptions of consequent configurations will be used to encode transitions.

The dataset \mathcal{D}_w consists of two blocks of facts. The first block introduces constants $c_1, \dots, c_{p(|w|)}$, for cells used in the computation, and defines the order in which they are located on the tape. Let ϱ be the interval $(-\infty, \infty)$; then \mathcal{D}_w contains the following facts, for all $i, j, k \in \mathbb{N}$ such that $1 \leq i, j \leq p(|w|)$, $i \neq j$, and $1 \leq k < p(|w|)$:

$$Cell^E(c_i) @ \varrho, Neq^E(c_i, c_j) @ \varrho, Next^E(c_k, c_{k+1}) @ \varrho,$$

where $Cell^E(c_i)$ states that c_i is a cell, $Neq^E(c_i, c_j)$ states that cell c_i is different from c_j , and $Next^E(c_k, c_{k+1})$ states that cell c_{k+1} is the right-neighbour of c_k on the tape (super-script E indicates EDB predicates). The second block of \mathcal{D}_w encodes the initial configuration, represented by constant q_0 , and contains the following facts for all pairs (c_i, w_i) , with w_i a constant representing the i -th symbol in w :

$$Head^E(c_1) @ \{1\}, State^E(q_0) @ \{1\}, Symb^E(c_i, w_i) @ \{1\},$$

where $Head^E(c)$ states that the head is over the cell c , $State^E(q)$ that the current state is q , and $Symb^E(c, s)$ that the cell c contains the symbol s .

The program Π_M transforms EDB atoms over P^E , for each P among $Head$, $State$, and $Symb$, into IDB atoms over P by means of rules $P(\tau) \leftarrow P^E(\tau)$, where τ is a

tuple of distinct variables of matching arity. The program also defines a predicate $R(x, y, u, z, v)$, representing the fact that in the previous configuration the state was x , the head was over a cell y containing a symbol u , and there was a cell z containing a symbol v . This is achieved by including the following rules in Π_M for every state q and for all alphabet symbols s, s' , where x, y, u, z, v are object variables:

$$\begin{aligned} Q(x, y, s, z, s') &\leftarrow \Box_{\{1\}} State(x) \wedge Cell^E(y) \wedge Cell^E(z), \\ Q(q, y, s, z, s') &\leftarrow \Box_{\{2\}} Head(y) \wedge Cell^E(z), \\ Q(q, y, u, z, s') &\leftarrow \Box_{\{3\}} Symb(y, u) \wedge Cell^E(z), \\ Q(q, y, s, z, v) &\leftarrow \Box_{\{4\}} Symb(z, v) \wedge Cell^E(y), \\ R(x, y, u, z, v) &\leftarrow \Box_{[1,4]} Q(x, y, u, z, v). \end{aligned}$$

The rules above exploit the gaps (of length 5) between time points encoding consequent configurations and the operator $\Box_{[1,4]}$ to simulate with R a conjunction of four IDB atoms. Then, each left-moving transition $(q, s) \mapsto (q', s', L)$ is encoded with the following rules:

$$\begin{aligned} State(q') \wedge Symb(y, s') &\leftarrow R(q, y, s, z, v), \\ Head(z) &\leftarrow R(x, y, u, z, v) \wedge Next^E(z, y), \\ Symb(z, v) &\leftarrow R(x, y, u, z, v) \wedge Neq^E(z, y), \end{aligned}$$

and each right-moving transition is encoded similarly. The first rule determines the new state and the contents of the cell under the head; the second rule provides the new position of the head; and the third states that all other cells remain unchanged. Finally, Π_M contains rule $\perp \leftarrow State(q)$ for each accepting state q , which ensures that Π_M and \mathcal{D}_w are \mathbb{Z} -inconsistent if and only if M accepts w . \square

It follows from Theorems 1 and 2 that the data complexity of \mathbb{Z} -consistency checking and \mathbb{Q} -consistency checking coincide for both DatalogMTL and DatalogMTL_{lin} ^{\Box} . As we will show in the remainder of this paper, however, \mathbb{Z} -consistency checking becomes significantly easier than \mathbb{Q} -consistency checking for a range of natural fragments of DatalogMTL.

4 Propositional Fragments

In this section we study data complexity of \mathbb{Z} -consistency checking in propositional fragments of DatalogMTL. It was shown by Brandt et al. (2018) that \mathbb{Q} -consistency checking in propositional DatalogMTL is P-hard, so inherently sequential and maybe intractable (the best known upper bound is PSPACE). We show that, in contrast, \mathbb{Z} -consistency checking is NC¹-complete, thus tractable and parallelisable.

It is worth to notice that propositional DatalogMTL under integer semantics is related to the LTL_{horn} ^{$\Box \circ$} fragment of Linear Temporal Logic (Artale et al. 2017), where formulas are in the Horn form and the only allowed operators are ‘everywhere in the future’ \Box and ‘in the next time point’ \circ . This fragment has been shown to be NC¹-complete in data complexity (Artale et al. 2015); however, the naive translation of DatalogMTL to LTL_{horn} ^{$\Box \circ$} , would result in an exponential increase of data complexity since facts in LTL_{horn} ^{$\Box \circ$} are punctual (rather than over intervals) and LTL_{horn} ^{$\Box \circ$} requires that, in

a dataset where t_1 and t_2 are the minimal and the maximal integers, all integers belonging to $[t_1, t_2]$ must occur explicitly in the dataset. It follows that known NC^1 upper bounds cannot be readily transferred to our setting.

In the remaining of this section we consider only integer semantics, so we fix the temporal domain \mathbb{T} to be the integers and thus all relevant notions are implicitly parametrised by \mathbb{Z} . We will also denote by Π an arbitrary propositional program and by \mathcal{D} a dataset, where Π is considered fixed for complexity analysis. We assume without loss of generality that \mathcal{D} uses only propositions mentioned in Π and the minimal integer mentioned in \mathcal{D} is 0; any dataset can be transformed into this form in AC^0 while preserving consistency with Π . We also assume that Π is normalised so that it does not use any nesting of temporal operators. Such normalisation may be achieved (in a data-independent way) by introducing fresh propositions for nested literals. We denote by $\text{prop}(\Pi)$ the set of propositions in Π and by $\text{lit}(\Pi)$ the set of all literals without nested temporal operators, with all propositions from $\text{prop}(\Pi)$, and with all integer endpoints bounded by the maximal integer in Π .

To show the NC^1 upper bound for \mathbb{Z} -consistency checking, we first construct an NFA $\mathcal{A}_{\Pi, \mathcal{D}}$ and a word $w_{\Pi, \mathcal{D}}$ (both dependent on Π and \mathcal{D}) such that Π and \mathcal{D} are \mathbb{Z} -consistent if and only if $\mathcal{A}_{\Pi, \mathcal{D}}$ accepts $w_{\Pi, \mathcal{D}}$. Although word acceptance by a fixed NFA is in NC^1 (Chandra, Stockmeyer, and Vishkin 1984), our construction does not yield the desired upper bound by the following two reasons: first, $w_{\Pi, \mathcal{D}}$ cannot be constructed in NC^1 (it is exponential in the size of \mathcal{D}) and, second, $\mathcal{A}_{\Pi, \mathcal{D}}$ depends on \mathcal{D} . To overcome the first, we exploit Chrobak's results on the normalisation of unary NFAs (Chrobak 1986) to construct a succinct NFA $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ and a word $\bar{w}_{\Pi, \mathcal{D}}$ such that $\mathcal{A}_{\Pi, \mathcal{D}}$ accepts $w_{\Pi, \mathcal{D}}$ if and only if $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ accepts $\bar{w}_{\Pi, \mathcal{D}}$. In this representation, $\bar{w}_{\Pi, \mathcal{D}}$ is of polynomial size in the size of \mathcal{D} and moreover can be constructed in NC^1 . To overcome the second reason, we observe that the only elements of $\mathcal{A}_{\Pi, \mathcal{D}}$ dependent on \mathcal{D} are the sets of initial and accepting states (which in turn are subsets of the data-independent set of all states), and this property carries over to the construction of the succinct automaton $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$. This suggests an NC^1 procedure where acceptance of $\bar{w}_{\Pi, \mathcal{D}}$ is checked in parallel for each automaton, the number of which is independent of the size of \mathcal{D} , obtained from a different choice of initial and final states.

We next move on to the construction of the automaton $\mathcal{A}_{\Pi, \mathcal{D}}$, and start by introducing the notions underpinning the definition of the automaton's states and transition function.

Definition 3. A set $q \subseteq \text{lit}(\Pi)$ of literals is Π -consistent if there is a model \mathfrak{M} of Π such that $\mathfrak{M}, 0 \models A$ if and only if $A \in q$, for each $A \in \text{lit}(\Pi)$. Then, let $\text{con}(\Pi)$ denote the set of all Π -consistent subsets of $\text{lit}(\Pi)$.

For $q, q' \in \text{con}(\Pi)$, q' is a Π -successor of q if there is a model \mathfrak{M} of Π such that

1. $\mathfrak{M}, 0 \models A$ if and only if $A \in q$, for each $A \in \text{lit}(\Pi)$; and
2. $\mathfrak{M}, 1 \models A'$ if and only if $A' \in q'$, for each $A' \in \text{lit}(\Pi)$.

Note that in the definition we use time points 0 and 1 just for definiteness, and, by the definition of a program, it would be equivalent to use t and $t + 1$ for any $t \in \mathbb{Z}$ instead.

Now, we define an automaton and input word, where Π -consistent set of literals constitutes states and a transition from q to q' is possible only if q' is a Π -successor of q .

Definition 4. The NFA $\mathcal{A}_{\Pi, \mathcal{D}} = (Q, \Sigma, \delta, Q_0, F)$ is as follows, where d is the maximal integer mentioned in \mathcal{D} :

- the set of states is $Q = \text{con}(\Pi)$;
- the alphabet is $\Sigma = \mathcal{P}(\text{prop}(\Pi))$;
- the transition function δ is such that

$$\delta(q, X) = \{q' \in Q \mid q' \text{ is a } \Pi\text{-successor of } q, X \subseteq q'\};$$

- set Q_0 of initial states contains all $q \in Q$ such that, for all $P \in \text{prop}(\Pi)$, if $\mathcal{D} \models P@(-\infty, -1]$ then $\Box_{[0, \infty)} P \in q$;
 - set F of final states contains all $q \in Q$ such that, for all $P \in \text{prop}(\Pi)$, if $\mathcal{D} \models P@[d + 1, \infty)$ then $\Box_{[0, \infty)} P \in q$.
- Moreover, $w_{\Pi, \mathcal{D}} = X_0 \cdots X_{d+1}$ is a word such that, for all $i \in \{0, \dots, d + 1\}$, $X_i = \{P \in \text{prop}(\Pi) \mid \mathcal{D} \models P@i\}$.

As established by the following lemma, our choice of states and transitions of $\mathcal{A}_{\Pi, \mathcal{D}}$ ensures that each run of the automaton (on any input word) corresponds to a model of Π .

Lemma 5. The following statements are equivalent for each sequence q_0, \dots, q_n of elements in $\text{con}(\Pi)$:

1. there exists a model \mathfrak{M} of Π such that $\mathfrak{M}, i \models A$ if and only if $A \in q_i$, for all $i \in \{0, \dots, n\}$ and $A \in \text{lit}(\Pi)$;
2. q_{i+1} is a Π -successor of q_i , for all $i \in \{0, \dots, n - 1\}$.

Proof sketch. Condition 1 implies condition 2 by the definition of a Π -successor. Next, we show the converse. Since q_0 and q_n are in $\text{con}(\Pi)$, there are models \mathfrak{M}_0 and \mathfrak{M}_n of Π such that $\mathfrak{M}_0, 0 \models A$ if and only if $A \in q_0$, for all $A \in \text{lit}(\Pi)$, and $\mathfrak{M}_n, n \models A'$ if and only if $A' \in q_n$, for all $A' \in \text{lit}(\Pi)$. Let interpretation \mathfrak{M} be as follows: for each $i \in \mathbb{Z}$ and $P \in \text{prop}(\Pi)$ let $\mathfrak{M}, i \models P$ if either $i < 0$ and $\mathfrak{M}_0, i \models P$, or $0 \leq i \leq n$ and $P \in q_i$, or $i > n$ and $\mathfrak{M}_n, i \models P$. We can show that, for each $i \in \{0, \dots, n\}$ and $A \in \text{lit}(\Pi)$, $\mathfrak{M}, i \models A$ if and only if $A \in q_i$. This allows us to show that \mathfrak{M} is a model of Π , so condition 1 holds. \square

Word $w_{\Pi, \mathcal{D}}$ in Definition 4 describes propositions entailed by \mathcal{D} at time points between 0 and d . Observe that since 0 and d are the minimal and the maximal integers in \mathcal{D} , if \mathcal{D} entails a proposition P at a time point smaller than 0 or greater than d , then $\mathcal{D} \models P@(-\infty, -1]$ or $\mathcal{D} \models P@[d + 1, \infty)$, respectively, which is reflected in the definition of initial and final states of $\mathcal{A}_{\Pi, \mathcal{D}}$. Using these observations and Lemma 5 we can then show that an accepting run q_{-1}, \dots, q_{d+1} of $\mathcal{A}_{\Pi, \mathcal{D}}$ on $w_{\Pi, \mathcal{D}}$ corresponds to a model of \mathcal{D} and Π , where q_i for a time point $i \in \{0, \dots, d\}$ describes propositions holding in i , whereas $q_{-1} \in Q_0$ and $q_{d+1} \in F$ are additional states describing propositions holding at (infinitely many) time points smaller than 0 and greater than d , respectively.

Lemma 6. Program Π and dataset \mathcal{D} are \mathbb{Z} -consistent if and only if $\mathcal{A}_{\Pi, \mathcal{D}}$ accepts $w_{\Pi, \mathcal{D}}$.

Proof. (\Rightarrow) Let \mathfrak{M} be a model of Π and \mathcal{D} and, for each integer i , let q_i be the (uniquely defined) set of literals $A \in \text{lit}(\Pi)$ satisfying $A \in q_i$ if and only if $\mathfrak{M}, i \models A$. By the definition of a Π -consistent set, we

have $q_i \in \text{con}(\Pi)$. Hence, for d the maximal integer in \mathcal{D} , we have $q_{-1}, \dots, q_{d+1} \in \text{con}(\Pi)$, and so by Lemma 5, q_{-1}, \dots, q_{d+1} is a sequence of Π -successors. Now, let $\mathcal{A}_{\Pi, \mathcal{D}} = (Q, \Sigma, \delta, Q_0, F)$ and $w_{\Pi, \mathcal{D}} = X_0 \dots X_{d+1}$ be as in Definition 4. We can show that $q_{-1} \in Q_0$, $q_{d+1} \in F$, and $X_i \subseteq q_i$ for all $i \in \{0, \dots, d\}$, so q_{-1}, \dots, q_{d+1} is an accepting run of $\mathcal{A}_{\Pi, \mathcal{D}}$ on $w_{\Pi, \mathcal{D}}$.

(\Leftarrow) Let q_{-1}, \dots, q_{d+1} be an accepting run of $\mathcal{A}_{\Pi, \mathcal{D}} = (Q, \Sigma, \delta, Q_0, F)$ on $w_{\Pi, \mathcal{D}} = X_0 \dots X_{d+1}$ (so d is the maximal integer in \mathcal{D}). Then, $q_{-1} \in Q_0$, $q_{d+1} \in F$, and q_{-1}, \dots, q_{d+1} is a sequence of Π -successors. By Lemma 5, there is a model \mathfrak{M} of Π such that $\mathfrak{M}, i \models A$ if and only if $A \in q_i$, for all $i \in \{-1, \dots, d+1\}$ and $A \in \text{lit}(\Pi)$. It remains to show that \mathfrak{M} is a model of \mathcal{D} . Assume that $\mathcal{D} \models P@i$, for some $P \in \text{prop}(\Pi)$ and $i \in \mathbb{Z}$. If $i < 0$, then $\mathcal{D} \models P@(-\infty, -1]$, since 0 is the minimal integer in \mathcal{D} . Hence, every $q \in Q_0$ contains $\Box_{[0, \infty)} P$, and so $\Box_{[0, \infty)} P \in q_{-1}$. Then, $\mathfrak{M}, -1 \models \Box_{[0, \infty)} P$, so $\mathfrak{M} \models P@(-\infty, -1]$, and thus $\mathfrak{M} \models P@i$. If $i > d$, then we can show in a similar way that $\mathcal{D} \models P@[d+1, \infty)$, so $\Box_{[0, \infty)} P \in q_{d+1}$, and thus $\mathfrak{M} \models P@[d+1, \infty)$. If $0 \leq i \leq d$, then $P \in q_i$, and so by the definition of \mathfrak{M} , we have $\mathfrak{M} \models P@i$. Thus, \mathfrak{M} is a model of Π and \mathcal{D} . \square

As already mentioned, Lemma 6 does not yield an NC^1 upper bound. Indeed, the length of the word $w_{\Pi, \mathcal{D}}$ is $d+2$, for d the maximal integer mentioned in \mathcal{D} ; and since integers in \mathcal{D} are encoded in binary, the length of $w_{\Pi, \mathcal{D}}$ is exponential in the size of \mathcal{D} and hence cannot be constructed in NC^1 . The exponential length is, however, only caused by long consecutive repetitions of some symbols; for example if $\mathcal{D} = \{P@(-\infty, 0], R@[0, d], S@[0, \infty)\}$, then $w_{\Pi, \mathcal{D}} = \{P, R, S\}\{R, S\} \dots \{R, S\}\{S\}$, where $\{R, S\}$ is repeated d times. We next show how to modify $\mathcal{A}_{\Pi, \mathcal{D}}$ and $w_{\Pi, \mathcal{D}}$ to obviate such repetitions and ensure that the new word can be constructed in NC^1 .

Our construction of the succinct automaton $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ and the word $\bar{w}_{\Pi, \mathcal{D}}$ relies on a key property of $\mathcal{A}_{\Pi, \mathcal{D}}$ and $w_{\Pi, \mathcal{D}}$: the (possibly infinitely many) lengths of runs of $\mathcal{A}_{\Pi, \mathcal{D}}$ from a state q to q' on words $X \dots X$ mentioning a single symbol X can be succinctly represented in a data-independent way. To show this property, we exploit a result by Chrobak (1986) on normalisation of unary automata.

Lemma 7. *For all $q, q' \in \text{con}(\Pi)$ and $X \subseteq \mathcal{P}(\text{prop}(\Pi))$ we can construct (without using \mathcal{D}) a finite set $S_{\Pi}(q, q', X)$ of pairs of non-negative integers such that the following statements are equivalent for each $\ell \in \mathbb{N}$:*

1. *there is a run of $\mathcal{A}_{\Pi, \mathcal{D}}$ from q to q' on word X^ℓ ;*
2. *$\ell = a + n \cdot b$ for some $(a, b) \in S_{\Pi}(q, q', X)$ and $n \in \mathbb{N}$.*

Proof. Let $q, q' \in \text{con}(\Pi)$ and $X \in \text{prop}(\Pi)$. Consider the NFA $\mathcal{A}_{\Pi, \mathcal{D}} = (Q, \Sigma, \delta, Q_0, F)$ from Definition 4 and a unary NFA $U_{\Pi} = (Q, \{X\}, \delta', q, q')$ with the same states Q , unary alphabet $\{X\}$, transition function δ' that is a projection of δ to X , and q and q' as the initial and final states. Since Q and δ in $\mathcal{A}_{\Pi, \mathcal{D}}$ do not depend on \mathcal{D} , so does automaton U_{Π} . Moreover, for every $\ell \in \mathbb{N}$, condition 1 holds if and only if U_{Π} accepts the word X^ℓ . Since U_{Π} is a unary NFA, by (Chrobak 1986, Lemma 4.3), there exists an equivalent

unary NFA U'_{Π} in Chrobak normal form, where the transition relation constitutes a path from the initial state followed by a single nondeterministic choice between several disjoint cycles. For each accepting state q_f in U'_{Π} let a_f and b_f be the lengths of the path from the initial state to q_f and of the cycle through q_f , respectively, and let $S_{\Pi}(q, q', X)$ be the set of pairs (a_f, b_f) for all accepting states q_f . So, for each $\ell \in \mathbb{N}$, U'_{Π} accepts X^ℓ if and only if condition 2 holds. \square

We are now ready to define the succinct automaton $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$, whose states coincide with those of $\mathcal{A}_{\Pi, \mathcal{D}}$; however, each alphabet symbol in $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ is now a pair (X, σ) consisting of a symbol X from the alphabet of $\mathcal{A}_{\Pi, \mathcal{D}}$ and a set σ of number pairs (a, b) . A transition from q to q' on symbol (X, σ) in $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ then corresponds to a run (of length ℓ) from q to q' on a word X^ℓ in $\mathcal{A}_{\Pi, \mathcal{D}}$, where $\ell = a + n \cdot b$ for some $(a, b) \in \sigma$ and $n \in \mathbb{N}$. This allows us also to define a succinct representation $\bar{w}_{\Pi, \mathcal{D}}$ of the word $w_{\Pi, \mathcal{D}}$.

Definition 8. *Let $\mathcal{A}_{\Pi, \mathcal{D}} = (Q, \Sigma, \delta, Q_0, F)$ be the automaton from Definition 4 and let*

$$\begin{aligned} \text{paths}(\Pi) &= \{(a, b) \mid (a, b) \in S_{\Pi}(q, q', X), \\ &\quad \text{for some } q, q' \in \text{con}(\Pi) \text{ and } X \subseteq \text{prop}(\Pi)\}. \end{aligned}$$

Then, $\bar{\mathcal{A}}_{\Pi, \mathcal{D}} = (Q, \bar{\Sigma}, \bar{\delta}, Q_0, F)$ is the NFA with

- *the alphabet $\bar{\Sigma} = \{(X, \sigma) \mid X \in \Sigma \text{ and } \sigma \subseteq \text{paths}(\Pi)\}$;*
- *the transition function $\bar{\delta}$ such that*

$$\bar{\delta}(q, (X, \sigma)) = \{q' \in Q \mid \sigma \cap S_{\Pi}(q, q', X) \neq \emptyset\}.$$

Let $t_1 < \dots < t_k$ be all integers mentioned in \mathcal{D} , and let $\varrho_1, \dots, \varrho_m$ be the subsequence of all non-empty intervals in the sequence $\{t_1\}, (t_1, t_2), \{t_2\}, \dots, (t_{k-1}, t_k), \{t_k\}, \{t_k + 1\}$. Then, let $\bar{w}_{\Pi, \mathcal{D}} = (X_1, \sigma_1) \dots (X_m, \sigma_m)$ be the $\bar{\Sigma}$ -word satisfying the following, for all $i \in \{1, \dots, m\}$:

- $X_i = \{P \in \text{prop}(\Pi) \mid \mathcal{D} \models P@_{\varrho_i}\}$;
- $\sigma_i = \{(a, b) \in \text{paths}(\Pi) \mid |\varrho_i| = a + n \cdot b \text{ for some } n \in \mathbb{N}\}$.

It is worth to recall that in this section we consider only \mathbb{Z} -intervals, which are sets of integers; so, for example, the interval $(2, 3)$ is empty and $|(2, 4)| = 1$. Next, observe that the length of $\bar{w}_{\Pi, \mathcal{D}}$ is linear in \mathcal{D} . In particular, for $\mathcal{D} = \{P@(-\infty, 0], R@[0, d], S@[0, \infty)\}$ as in our previous example, we have $\varrho_1 = \{0\}$, $\varrho_2 = (0, d)$, $\varrho_3 = \{d\}$, $\varrho_4 = \{d+1\}$, and

$$\bar{w}_{\Pi, \mathcal{D}} = (\{P, R, S\}, \sigma_1)(\{R, S\}, \sigma_2)(\{R, S\}, \sigma_3)(\{S\}, \sigma_4),$$

for the appropriate $\sigma_i \subseteq \text{Paths}(\Pi)$. Indeed, the $(d-1)$ times repetition of $\{R, S\}$ in $w_{\Pi, \mathcal{D}}$ expressing that R and S hold everywhere in $(0, d)$ has been replaced in $\bar{w}_{\Pi, \mathcal{D}}$ with a single element $(\{R, S\}, \sigma_2)$, where σ_2 contains a pair (a, b) such that $|(0, d)| = a + n \cdot b$ for some $n \in \mathbb{N}$.

Lemma 9. *NFA $\mathcal{A}_{\Pi, \mathcal{D}}$ accepts word $w_{\Pi, \mathcal{D}}$ if and only if NFA $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ accepts word $\bar{w}_{\Pi, \mathcal{D}}$.*

Proof. Let us fix notation where d is the maximal integer in \mathcal{D} ; $t_1 < \dots < t_k$ are all integers in \mathcal{D} (so $t_1 = 0$ and $t_k = d$); $\varrho_1, \dots, \varrho_m$ is the subsequence of all non-empty intervals in $\{t_1\}, (t_1, t_2), \{t_2\}, \dots, (t_{k-1}, t_k), \{t_k\}, \{t_k + 1\}$

(so $\varrho_1 = \{0\}$ and $\varrho_m = \{d+1\}$); $w_{\Pi, \mathcal{D}} = X_0 \cdots X_{d+1}$; $\bar{w}_{\Pi, \mathcal{D}} = (\bar{X}_1, \sigma_1) \cdots (\bar{X}_m, \sigma_m)$; $\mathcal{A}_{\Pi, \mathcal{D}} = (Q, \Sigma, \delta, Q_0, F)$; and $\bar{\mathcal{A}}_{\Pi, \mathcal{D}} = (Q, \bar{\Sigma}, \bar{\delta}, Q_0, F)$. We will denote the minimal and the maximal integers contained in a bounded interval ϱ by $\lfloor \varrho \rfloor$ and $\lceil \varrho \rceil$, respectively. Note that since the endpoints of intervals $\varrho_1, \dots, \varrho_m$ coincide with integers mentioned in \mathcal{D} , we can show that $\mathcal{D} \models P @ \{t\}$ for a proposition P and $t \in \varrho_i$ imply $\mathcal{D} \models P @ \varrho_i$. Thus, by the definition of $w_{\Pi, \mathcal{D}}$, we have $X_{\lfloor \varrho_i \rfloor} = X_{\lfloor \varrho_i \rfloor + 1} = \dots = X_{\lceil \varrho_i \rceil}$ and so, by the definition of $\bar{w}_{\Pi, \mathcal{D}}$, all these symbols are equal to \bar{X}_i .

(\Rightarrow) Let q_{-1}, \dots, q_{d+1} be an accepting run of $\mathcal{A}_{\Pi, \mathcal{D}}$ on $w_{\Pi, \mathcal{D}}$. We show that $r = q_{-1}, q_{\lceil \varrho_1 \rceil}, \dots, q_{\lceil \varrho_m \rceil}$ is an accepting run of $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ on $\bar{w}_{\Pi, \mathcal{D}}$. Since $q_{\lceil \varrho_m \rceil} = q_{d+1}$, and $\mathcal{A}_{\Pi, \mathcal{D}}$ and $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ have the same sets of initial and accepting states, if r is a run then it is an accepting run. To show that r is a run we need to show that for q and $q_{\lceil \varrho_s \rceil}$ consecutive states in r , we have that $q_{\lceil \varrho_s \rceil} \in \bar{\delta}(q, (\bar{X}_s, \sigma_s))$. Let $i = \lfloor \varrho_s \rfloor - 1$ and $j = \lceil \varrho_s \rceil$ (so $q = q_i$ and $q_{\lceil \varrho_s \rceil} = q_j$) hence, as we have already observed, $X_{i+1} = \dots = X_j = \bar{X}_s$. Since q_{-1}, \dots, q_{d+1} is an accepting run of $\mathcal{A}_{\Pi, \mathcal{D}}$ on $w_{\Pi, \mathcal{D}}$, it contains a run of length $j-i$ from q_i to q_j on $X_{i+1} \cdots X_j$. Thus, by Lemma 7, there is a pair $(a, b) \in S_{\Pi}(q_i, q_j, \bar{X}_s)$ such that $a + n \cdot b = j - i$ for some $n \in \mathbb{N}$. By the definition of $\bar{w}_{\Pi, \mathcal{D}}$ and the fact that $|\varrho_s| = j - i$, we obtain $(a, b) \in \sigma_s$. Thus, $\sigma_s \cap S_{\Pi}(q_i, q_j, \bar{X}_s) \neq \emptyset$, and so, by the definition of $\bar{\delta}$, we have $q_j \in \bar{\delta}(q_i, (\bar{X}_s, \sigma_s))$. Thus, $q_{\lceil \varrho_s \rceil} \in \bar{\delta}(q, (\bar{X}_s, \sigma_s))$.

(\Leftarrow) Let q_{-1}, q_0, \dots, q_m be an accepting run of $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ on $\bar{w}_{\Pi, \mathcal{D}}$; we show existence of an accepting run of $\mathcal{A}_{\Pi, \mathcal{D}}$ on $w_{\Pi, \mathcal{D}}$. Recall that $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ and $\mathcal{A}_{\Pi, \mathcal{D}}$ have the same initial and final states, and $X_{\lfloor \varrho_s \rfloor} = \dots = X_{\lceil \varrho_s \rceil} = \bar{X}_s$ for every $s \in \{1, \dots, m\}$, so it suffices to show that $q_s \in \bar{\delta}(q_{s-1}, (\bar{X}_s, \sigma_s))$ implies existence of a run of $\mathcal{A}_{\Pi, \mathcal{D}}$ (of length $|\varrho_s| + 1$) from q_{s-1} to q_s on the word $\bar{X}_s^{|\varrho_s|}$. Since $q_s \in \bar{\delta}(q_{s-1}, (\bar{X}_s, \sigma_s))$, by the definition of $\bar{\delta}$, there is a pair $(a, b) \in \sigma_s \cap S_{\Pi}(q_{s-1}, q_s, \bar{X}_s)$. Hence, $(a, b) \in \sigma_s$ and so, by the definition of $\bar{w}_{\Pi, \mathcal{D}}$, there is $n \in \mathbb{N}$ such that $|\varrho_s| = a + n \cdot b$. Thus, by Lemma 7 and the fact that $(a, b) \in S_{\Pi}(q_{s-1}, q_s, \bar{X}_s)$, there is a run of $\mathcal{A}_{\Pi, \mathcal{D}}$ from q_{s-1} to q_s on the word $\bar{X}_s^{|\varrho_s|}$. \square

We are ready to show the main result of this section.

Theorem 10. *Checking \mathbb{Z} -consistency in propositional DatalogMTL is in NC^1 in data complexity.*

Proof. Fix a propositional DatalogMTL program Π and consider an input dataset \mathcal{D} . By Lemmas 6 and 9, it suffices to check whether $\bar{\mathcal{A}}_{\Pi, \mathcal{D}} = (Q, \bar{\Sigma}, \bar{\delta}, Q_0, F)$ accepts $\bar{w}_{\Pi, \mathcal{D}} = (\bar{X}_1, \sigma_1) \cdots (\bar{X}_m, \sigma_m)$. We first argue that $\bar{w}_{\Pi, \mathcal{D}}$ can be constructed in TC^0 in the size of \mathcal{D} . For this, compute the sequence $t_1 < \dots < t_n$ of integers mentioned in \mathcal{D} , where we note that sorting integers is feasible in TC^0 (Chandra, Stockmeyer, and Vishkin 1984). Then, it is straightforward to construct, in TC^0 , intervals $\varrho_1, \dots, \varrho_m$ and sets $\bar{X}_1, \dots, \bar{X}_m$ of propositions occurring in the definition of $\bar{w}_{\Pi, \mathcal{D}}$. It remains to compute σ_i , for every $i \in \{1, \dots, m\}$.

To this end, we compute $|\varrho_i|$, which can be done in TC^0 , and then list all $(a, b) \in \text{paths}(\Pi)$ such that $|\varrho_i| = a + n \cdot b$, for some $n \in \mathbb{N}$, which can be done independently from \mathcal{D} . Hence, the construction of $\bar{w}_{\Pi, \mathcal{D}}$ is in TC^0 .

Finally, we will use the fact that the acceptance problem for a fixed NFA is in NC^1 (Holzer and Kutrib 2011). Although the sets of initial and final states in $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ depend on \mathcal{D} , the set of all states of $\bar{\mathcal{A}}_{\Pi, \mathcal{D}}$ does not. As a result, there is a constant (in the size of \mathcal{D}) number of automata of the form $(Q, \bar{\Sigma}, \bar{\delta}, Q'_0, F')$ with $Q'_0, F' \subseteq Q$. We can thus check in parallel (in NC^1 since each automaton is fixed) which automata accept $\bar{w}_{\Pi, \mathcal{D}}$. Then, we can compute in TC^0 the sets Q_0 and F and verify whether, according to the former NC^1 procedure, the automaton $(Q, \bar{\Sigma}, \bar{\delta}, Q_0, F)$ accepts $\bar{w}_{\Pi, \mathcal{D}}$. The overall procedure is in NC^1 in the size of \mathcal{D} . \square

Our next theorem provides the matching NC^1 lower bound applicable to different fragments of propositional DatalogMTL. The bound is shown by reduction of the word acceptance problem for a fixed DFA, which is NC^1 -hard due to the existence of NC^1 -complete regular languages (Barrington et al. 1992).

Theorem 11. *Checking \mathbb{Z} -consistency is NC^1 -hard in data complexity for the propositional fragments of DatalogMTL $_{\text{lin}}^{\diamond}$ and DatalogMTL $_{\text{core}}^{\square}$.*

Proof. We start with the case of DatalogMTL $_{\text{lin}}^{\diamond}$. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, q_f)$ be a fixed DFA and let $w = w_0 \cdots w_{\ell}$ be an input word of length $\ell + 1$. Assume without loss of generality that $q_0 \notin \delta(q, \sigma)$ for each $q \in Q$ and $\sigma \in \Sigma$. We can use a similar idea as in (Artale et al. 2015) to construct a propositional DatalogMTL $_{\text{lin}}^{\diamond}$ program $\Pi_{\mathcal{A}}$ independent on w and a dataset \mathcal{D}_w such that the construction of \mathcal{D}_w is in AC^0 in the size of w and \mathcal{A} accepts w if and only if $\Pi_{\mathcal{A}}$ and \mathcal{D}_w are \mathbb{Z} -inconsistent. This yields the required bound since NC^1 is closed under complementation.

Let R_{q_0} and P_a , for each $a \in \Sigma$, be EDB propositions, and let R_q , for each $q \in Q \setminus \{q_0\}$, be an IDB proposition. We define

$$\mathcal{D}_w = R_{q_0} @ \{0\} \cup \{P_a @ \{j\} \mid w_j = a \text{ and } j \in \{0, \dots, \ell\}\}.$$

Then, $\Pi_{\mathcal{A}}$ is a program consisting of the following linear rules, where we note that R_{q_0} does not occur in the head of a rule due to our assumption on q_0 :

$$\begin{aligned} R_{q'} &\leftarrow \diamond_{\{1\}} R_q \wedge \diamond_{\{1\}} P_a, \quad \text{whenever } q' = \delta(q, a), \\ \perp &\leftarrow R_{q_f}. \end{aligned}$$

It can be shown that $\Pi_{\mathcal{A}}$ and \mathcal{D}_w are \mathbb{Z} -inconsistent if and only if \mathcal{A} accepts w .

Consider now DatalogMTL $_{\text{core}}^{\square}$. Given \mathcal{A} and w as above, we construct a propositional DatalogMTL $_{\text{core}}^{\square}$ program $\Pi'_{\mathcal{A}}$ and a dataset \mathcal{D}'_w . For each $a \in \Sigma$ and $q \in Q$ consider an EDB proposition P_{qa} and IDB proposition R_{qa} . Then, let

$$\begin{aligned} \mathcal{D}'_w &= \{P_{q_0 a} @ \{0\} \mid a \in \Sigma\} \cup \\ &\quad \{P_{qa} @ \{3j+1\} \mid w_j = a, q \in Q, j \in \{0, \dots, \ell\}\}, \end{aligned}$$

and program Π'_A consist of the following rules:

$$\begin{aligned} R_{qa} &\leftarrow P_{qa}, & \text{for each } q \in Q \text{ and } a \in \Sigma, \\ R_{q'a'} &\leftarrow \boxminus_{[2,3]} R_{qa}, & \text{whenever } q' = \delta(q, a) \text{ and } a' \in \Sigma, \\ \perp &\leftarrow R_{qfa}, & \text{for every } a \in \Sigma. \end{aligned}$$

Intuitively, time points of the form $3j + 1$ are used to encode the input word; if the j -th symbol of w is a , then $\mathcal{D}'_w \models P_{qa} @ \{3j + 1\}$, for all $q \in Q$. Time points of the form $3j$ encode states in the run of \mathcal{A} on w ; if q is the j -th state in the run, then $(\Pi'_A, \mathcal{D}'_w) \models R_{qa} @ \{3j\}$, for all $a \in \Sigma$. Then, literal $\boxminus_{[2,3]} R_{qa}$ simulates a conjunction by stating that the previous state in the run is q and the most recent input symbol is a , which allows us to determine the next state in the run. As a result, Π'_A and \mathcal{D}'_w are \mathbb{Z} -inconsistent if and only if \mathcal{A} accepts w . \square

Theorems 10 and 11 imply tight complexity bounds for a wide range of propositional fragments of DatalogMTL.

Corollary 12. *Checking \mathbb{Z} -consistency is NC^1 -complete in data complexity for the propositional fragments of each of the following languages: DatalogMTL, DatalogMTL_{lin}, DatalogMTL_{lin}[◇], DatalogMTL_{lin}[◇], DatalogMTL_{core}, and DatalogMTL_{core}[◇].*

Furthermore, the only metric operator used in the reduction for DatalogMTL_{lin}[◇] is the punctual past diamond operator $\diamond_{\{1\}}$; thus, same reduction can be used to show that consistency checking for the linear fragment of propositional Datalog_{IS} is NC^1 -hard in data complexity. Note that Theorem 10 implies the matching upper bound for such a fragment of Datalog_{IS}, and so it is NC^1 -complete.

We conclude this section by observing that the propositional fragment of DatalogMTL_{core}[◇] is not covered by Corollary 12. We will show in the next section that \mathbb{Z} -consistency checking in this fragment is below NC^1 (and hence the same applies to the propositional core fragment of forward-propagating Datalog_{IS}).

5 Core Fragment of Low Complexity

In this section we show that \mathbb{Z} -consistency checking for DatalogMTL_{core}[◇] is in ACC^0 . This is in contrast to \mathbb{Q} -consistency checking in this fragment, which can be easily shown TC^0 -hard already for the propositional case; indeed, a simple reduction from the TC^0 -complete problem of checking whether $a \cdot b = c$ for integers a, b, c in binary (Hesse 2001) to \mathbb{Q} -inconsistency can be achieved using the program $\{P \wedge R \rightarrow \perp\}$, with propositions P and R , and the dataset $\{P @ \{a\}, R @ \{\frac{c}{b}\}\}$. We will also prove that \mathbb{Z} -consistency in propositional DatalogMTL_{core}[◇] is not in AC^0 . In the rest of the section we assume integer semantics only.

Similarly to the previous section, we start with a normal form. A DatalogMTL_{core}[◇] program is *normal* if it has only rules of the following forms, for $\alpha, \alpha_1, \alpha_2$, and β atoms, and ϱ a non-empty bounded positive interval of the form $[t_1, t_2]$:

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

Normalisation yields a program that is a conservative extension of the original one: nested operators are flattened

using fresh predicates, open intervals are modified as in Section 4, rules without metric operators are rewritten with dummy $\diamond_{\{0\}}$, and each rule $P(\tau) \leftarrow \diamond_{[t, \infty)} P'(\tau')$ is replaced by rules $Q(\tau) \leftarrow \diamond_{\{t\}} P'(\tau')$, $Q(\tau) \leftarrow \diamond_{\{1\}} Q(\tau)$ and $P(\tau) \leftarrow \diamond_{\{0\}} Q(\tau)$ with Q a fresh predicate of the same arity as P . Hence, in the rest of this section we concentrate on normal DatalogMTL_{core}[◇] programs.

Now, we concentrate on the propositional fragment of DatalogMTL_{core}[◇]—as we will show later, our approach can be easily extended to non-propositional case, as well. In particular, we characterise fact entailment for a propositional normal program Π in terms of existence of a specific path in the graph corresponding to Π as defined next.

Definition 13. *For each propositional DatalogMTL_{core}[◇] program Π in normal form, G_Π is the directed edge-weighted multigraph with a vertex v_P for each proposition P in Π , and an edge from v_P to v_Q of weight t for each rule $Q \leftarrow \diamond_{\varrho} P$ in Π and each integer $t \in \varrho$.*

In what follows, it is convenient to distinguish *positive* rules and programs—that is, those that do not mention \perp . Our next lemma establishes that, if Π is positive, then paths in G_Π concisely represent all the derivations in Π .

Lemma 14. *The following are equivalent for each propositional positive DatalogMTL_{core}[◇] program Π in normal form, dataset \mathcal{D} , and punctual fact $Q @ \{t\}$:*

- $(\Pi, \mathcal{D}) \models Q @ \{t\}$;
- *there is a fact $P @ \{t'\}$ with $\mathcal{D} \models P @ \{t'\}$ or rule $P \leftarrow \top$ in Π , and a path of length $t - t'$ from v_P to v_Q in G_Π .*

The next lemma guarantees that weights of all paths between two nodes in G_Π can be concisely represented. This lemma follows from the normal form for unary NFAs by Chrobak (1986), which we have already exploited in the previous section (see Lemma 7).

Lemma 15. *Let G be a directed edge-weighted multigraph with non-negative integer weights and let v_1, v_2 be vertices in G . Then, it is possible to construct a finite set $S_G(v_1, v_2)$ of pairs of non-negative integers, such that there is a path in G of weight ℓ from v_1 to v_2 if and only if $\ell = a + n \cdot b$, for some $(a, b) \in S_G(v_1, v_2)$ and $n \in \mathbb{N}$.*

Let Π be a propositional positive \top -free DatalogMTL_{core}[◇] program and a \mathcal{D} be dataset with all intervals of the form $[t_1, t_2]$ for $t_1, t_2 \in \mathbb{Z}$. Lemmas 14 and 15 suggest that to check if Π and \mathcal{D} entail a fact $Q @ \{t\}$, it suffices to find a fact $P @ [t_1, t_2]$ in \mathcal{D} , a pair (a, b) in $S_{G_\Pi}(v_P, v_Q)$, and an integer n such that

$$t_1 + a + n \cdot b \leq t \leq t_2 + a + n \cdot b.$$

This result can be easily extended to rules with \top and open or unbounded intervals in datasets and generalised to non-propositional programs via standard grounding.

We can exploit these results for punctual fact entailment to devise an algorithm for inconsistency checking. In particular, a propositional DatalogMTL_{core}[◇] program Π in normal form and a dataset \mathcal{D} are inconsistent if and only if there is a rule $\perp \leftarrow Q_1 \wedge Q_2$ in Π and a time point t such that both Q_1 and Q_2 hold at t in the least model of \mathcal{D} and the

positive subset of Π . Although existence and uniqueness of such model is ensured by the results in Wałęga et al. (2019), the number of candidate time points t is unbounded. Theorem 17 resolves this difficulty, and shows that \mathbb{Z} -consistency checking is in ACC^0 . Our bound relies on the following observation on the complexity of integer division.¹

Lemma 16. *Let c be a positive integer. Then checking if c divides an integer (given in binary) can be done in $\text{AC}^0[c]$.*

Proof. For each integer m of length k we have that

$$m \equiv m_k \cdot r_k + \dots + n_1 \cdot r_1 \pmod{c},$$

where m_i is the i -th bit in the binary representation of m and r_i is the remainder of dividing 2^i by c . Since $r_i < c$ for all i and $r_i = r_j$ implies $r_{i+1} = r_{j+1}$ for all i and j , we can, for each i , construct (in DLOGTIME) a constant circuitry! computing r_i in unary. Multiplication $m_i \cdot r_i$ can be realised by i binary *AND* gates, and the results for all i can be then aggregated in one *MOD* _{c} gate to get the final answer. \square

We are ready to state the main result of this section, which implies ACC^0 membership of consistency checking.

Theorem 17. *For every DatalogMTL_{core}[◇] program Π there exists a finite set of integers $\{c_1, \dots, c_k\}$ such that checking \mathbb{Z} -consistency of Π and a dataset \mathcal{D} is in $\text{AC}^0[c_1, \dots, c_k]$.*

Proof sketch. First, assume that Π is a propositional normal \top -free DatalogMTL_{core}[◇] program and \mathcal{D} is a dataset with all intervals of the form $[t_1, t_2]$ with $t_1, t_2 \in \mathbb{Z}$. Since all derivations in DatalogMTL_{core}[◇] are linear, Π and \mathcal{D} are inconsistent if and only if there is a rule $\perp \leftarrow Q_1 \wedge Q_2$ in Π and two facts $P_1 @ [t_1^L, t_1^R], P_2 @ [t_2^L, t_2^R]$ in \mathcal{D} such that $(\Pi^+, \{P_i @ [t_i^L, t_i^R]\}) \models Q_i @ \{t\}$ for both i and some time point t , where Π^+ is the set of all positive rules of Π . Each triple of a rule and two facts can be checked in parallel, so in the rest of the proof we assume that such a triple is fixed. Thus, we can use Lemma 14 to check consistency by verifying existence of two paths in G_Π , one from v_{P_1} to v_{Q_1} of weight ℓ_1 and another from v_{P_2} to v_{Q_2} of weight ℓ_2 , with

$$t_1^L - t_2^R \leq \ell_1 - \ell_2 \leq t_2^L - t_1^R. \quad (3)$$

By Lemma 15, we can represent, for both i , the weights of paths from v_{P_i} to v_{Q_i} with a finite set $S_{G_\Pi}(v_{P_i}, v_{Q_i})$ of pairs of integers. So, to check (3) it suffices to verify if there are $(a_1, b_1) \in S_{G_\Pi}(v_{P_1}, v_{Q_1})$, $(a_2, b_2) \in S_{G_\Pi}(v_{P_2}, v_{Q_2})$, and positive integers n_1, n_2 such that

$$t_1^L - t_2^R \leq (a_1 + n_1 \cdot b_1) - (a_2 + n_2 \cdot b_2) \leq t_2^L - t_1^R,$$

which, by Bézout's identity, holds if and only if

$$\left\lfloor \frac{t_1^L - t_2^R - a_1 + a_2}{c} \right\rfloor \neq \left\lfloor \frac{t_2^L - t_1^R - a_1 + a_2}{c} \right\rfloor, \quad (4)$$

where c is the greatest common divisor of b_1 and b_2 . Note that b_1 and b_2 do not depend on \mathcal{D} , so we can apply Lemma 16 and verify (4) in $\text{AC}^0[c]$ by checking whether there is an integer between $t_1^L - t_2^R$ and $t_2^L - t_1^R$ divided by c

¹We thank Rahul Santhanam for proving Lemma 16.

(note that for this it is enough to check at most c consecutive numbers in this interval). So, the overall consistency can be checked in $\text{AC}^0[c_1, \dots, c_k]$, where c_1, \dots, c_k are the greatest common divisors of all possible b_1 and b_2 of G_Π . The general case with \top in Π and unbounded intervals in \mathcal{D} can be handled in a similar way.

Finally, assume now that Π is a non-propositional normal program. Although the grounding $\Pi_{\mathcal{D}}$ of Π with constants from a dataset \mathcal{D} can be computed in AC^0 , the graph $G_{\Pi_{\mathcal{D}}}$ depends not only on Π but also on \mathcal{D} , which means that the application of the above procedure to $G_{\Pi_{\mathcal{D}}}$ does not immediately imply $\text{AC}^0[c_1, \dots, c_k]$ complexity (for any c_1, \dots, c_k). We can observe, however, that if two datasets \mathcal{D}_1 and \mathcal{D}_2 are the same modulo renaming constants not occurring in Π , then $G_{\Pi_{\mathcal{D}_1}}$ and $G_{\Pi_{\mathcal{D}_2}}$ are isomorphic. Since we are essentially interested in datasets with at most two facts and with arity of predicates fixed by Π , the number of non-isomorphic graphs for such datasets can be bounded without knowing \mathcal{D} ; thus, we can compute representatives of such graphs in a data-independent way and reuse our procedure for ground programs with the same complexity. \square

We conclude this section with the observation that the bound of Theorem 17 cannot be significantly improved, even for propositional DatalogMTL_{core}[◇] programs.

Theorem 18. *Checking \mathbb{Z} -consistency in propositional DatalogMTL_{core}[◇] program is not in AC^0 .*

Proof sketch. We prove the theorem by showing that for every odd prime integer c there exists a propositional DatalogMTL_{core}[◇] program Π such that the *MOD* _{c} problem—that is, the problem of checking whether c divides the number of true bits in an input—is reducible to consistency for Π . This is enough for the theorem, since *MOD* _{c} problem is not in AC^0 by the Razborov-Smolensky theorem (Razborov 1987; Smolensky 1987).

Given an odd prime c , we consider the following program, for propositions P, P' , and R :

$$\Pi = \{P' \leftarrow P, P' \leftarrow \Diamond_{\{c\}} P', \perp \leftarrow P' \wedge R\}.$$

It is folklore that *MOD* _{c} is true for an input if and only if c divides the number d , the binary representation of which has 1 only in the bits with numbers $(c-1) \cdot i$ and such that the i -th input bit is true (this immediately follows from Fermat's little theorem, claiming that $a^{c-1} \equiv 1 \pmod{c}$ for all prime c and all integers a , including 2). So, we can reduce each such input to the dataset $\mathcal{D} = \{P @ \{0\}, R @ \{d\}\}$, with d defined as above. Then, Π and \mathcal{D} are inconsistent if and only if c divides d —that is, if and only if *MOD* _{c} holds. \square

6 Conclusions

We have shown that the data complexity of reasoning in DatalogMTL and its fragments is not harder over integers than over rationals; and although in full DatalogMTL both semantics yield PSPACE-completeness, there are interesting fragments of DatalogMTL in which choosing the integer semantics makes reasoning significantly easier. In particular, we have proved that adopting the integer instead of rational semantics, results in a drop of the data complexity from

P-hard to NC^1 -complete for the propositional fragment of DatalogMTL; and from TC^0 -hard to ACC^0 , for (both propositional and non-propositional) DatalogMTL $_{core}^\diamond$. It is worth mentioning that similar fragments of DatalogMTL were studied under the pointwise semantics (called also as the event-based semantics), where a timeline contains only time points explicitly mentioned in a dataset (Ryzhikov, Wałęga, and Zakharyashev 2019). In this setting, the data complexity of reasoning was shown to be P-complete for propositional DatalogMTL and NL-complete for propositional DatalogMTL $_{core}^\diamond$; so in both cases higher than under integer semantics. As a future work we plan to construct practical algorithms for reasoning in the low complexity fragments of DatalogMTL under integer semantics and consider their applications to stream reasoning.

Acknowledgments

The authors thank Rahul Santhanam for proving Lemma 16. 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), OASIS (EP/S032347/1) and UK FIRES (EP/S019111/1).

References

- Artale, A.; Kontchakov, R.; Kovtunova, A.; Ryzhikov, V.; Wolter, F.; and Zakharyashev, M. 2015. First-order rewritability of temporal ontology-mediated queries. In *IJCAI*, 2706–2712. AAAI Press.
- Artale, A.; Kontchakov, R.; Kovtunova, A.; Ryzhikov, V.; Wolter, F.; and Zakharyashev, M. 2017. Ontology-mediated query answering over temporal data: A survey. In *TIME*.
- Baader, F.; Borgwardt, S.; Koopmann, P.; Ozaki, A.; and Thost, V. 2017. Metric temporal description logics with interval-rigid names. In *FroCoS*, 60–76.
- Barrington, D. A. M.; Compton, K.; Straubing, H.; and Thérien, D. 1992. Regular languages in NC^1 . *J. Comput. Syst. Sci.* 44(3):478–499.
- Brandt, S.; Kontchakov, R.; Ryzhikov, V.; Xiao, G.; and Zakharyashev, M. 2017. Ontology-based data access with a Horn fragment of metric temporal logic. In *AAAI*, 1070–1076.
- Brandt, S.; Kalaycı, E. G.; Ryzhikov, V.; Xiao, G.; and Zakharyashev, M. 2018. Querying log data with metric temporal logic. *J. Artif. Intell. Res.* 62:829–877.
- Brzowska, C. 1998. Programming in metric temporal logic. *Theoretical computer science* 202(1-2):55–125.
- Chandra, A. K.; Stockmeyer, L.; and Vishkin, U. 1984. Constant depth reducibility. *SIAM J. Comput.* 13(2):423–439.
- Chomicki, J., and Imieliński, T. 1988. Temporal deductive databases and infinite objects. In *PODS*, 61–73.
- Chrobak, M. 1986. Finite automata and unary languages. *Theor. Comput. Sci.* 47:149–158.
- Doherty, P.; Kvarnström, J.; and Heintz, F. 2009. A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems. *Auton. Agents Multi-Agent Syst.* 19(3):332–377.
- Gutiérrez Basulto, V.; Jung, J. C.; and Kontchakov, R. 2016. Temporalized EL ontologies for accessing temporal data: Complexity of atomic queries. 1102–1108.
- Gutiérrez-Basulto, V.; Jung, J. C.; and Ozaki, A. 2016. On metric temporal description logics. In *ECAI*, 837–845.
- Hesse, W. 2001. Division is in uniform TC^0 . In *ICALP*, 104–114. Springer.
- Holzer, M., and Kutrib, M. 2011. Descriptive and computational complexity of finite automata—a survey. *Information and Computation* 209(3):456–470.
- Koymans, R. 1990. Specifying real-time properties with metric temporal logic. *Real-Time Systems* 2(4):255–299.
- Ouaknine, J., and Worrell, J. 2008. Some recent results in metric temporal logic. In *FORMATS*, 1–13.
- Papadimitriou, C. H. 2003. *Computational complexity*. John Wiley and Sons Ltd.
- Razborov, A. A. 1987. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Math. Notes* 41(4):333–338.
- Ryzhikov, V.; Wałęga, P. A.; and Zakharyashev, M. 2019. Data complexity and rewritability of ontology-mediated queries in metric temporal logic under the event-based semantics. In *IJCAI*, 1851–1857.
- Smolensky, R. 1987. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *STOC*, 77–82.
- Wałęga, P. A.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2019. DatalogMTL: Computational complexity and expressive power. In *IJCAI*, 1886–1892.
- Wałęga, P.; Cuenca Grau, B.; and Kaminski, M. 2019. Reasoning over streaming data in metric temporal datalog. In *AAAI*, 1941–1948.