Journal of Artificial Intelligence Research 76 (2023) 471–521 Submitted 07/2022; published 01/2023

Finite Materialisability of Datalog Programs with Metric Temporal Operators

Przemysław Andrzej Wałęga PRZEMYSLAW.WALEGA@CS.OX.AC.UK
Michał Zawidzki MICHAL.ZAWIDZKI@CS.OX.AC.UK
Bernardo Cuenca Grau BERNARDO.CUENCA.GRAU@CS.OX.AC.UK
Department of Computer Science
University of Oxford
Parks Rd, Oxford, OX1 3QD, UK

Abstract

DatalogMTL is an extension of Datalog with metric temporal operators that has recently found applications in stream reasoning and temporal ontology-based data access. In contrast to plain Datalog, where materialisation (a.k.a. forward chaining) naturally terminates in finitely many steps, reaching a fixpoint in DatalogMTL may require infinitely many rounds of rule applications. As a result, existing reasoning systems resort to other approaches, such as constructing large Büchi automata, whose implementations turn out to be highly inefficient in practice.

In this paper, we propose and study finitely materialisable DatalogMTL programs, for which forward chaining reasoning is guaranteed to terminate. We consider a data-dependent notion of finite materialisability of a program, where termination is guaranteed for a given dataset, as well as a data-independent notion, where termination is guaranteed regardless of the dataset. We show that, for bounded programs (a natural DatalogMTL fragment for which reasoning is as hard as in the full language), checking data-dependent finite materialisability is ExpSpace-complete in combined complexity and PSpace-complete in data complexity; furthermore, we propose a practical materialisation-based decision procedure that works in doubly exponential time. We show that checking data-independent finite materialisability for bounded progams is computationally easier, namely ExpTime-complete; moreover, we propose sufficient conditions for data-indenpendent finite materialisability that can be efficiently checked. We provide also the complexity landscape of fact entailment for different classes of finitely materialisable programs; surprisingly, we could identify a large class of finitely materialisable programs, called MTL-acyclic programs, for which fact entailment has exactly the same data and combined complexity as in plain Datalog, which makes this fragment especially well suited for big-scale applications.

1. Introduction

DatalogMTL is a temporal extension of Datalog where atoms in rules may include operators from metric temporal logic (MTL) interpreted over the rational timeline (Brandt et al., 2018). DatalogMTL is a powerful temporal rule language, which has found applications in stream reasoning (Wałęga et al., 2019) and temporal ontology-based data access (Artale et al., 2017; Kikot et al., 2018), amongst others (Nissl & Sallinger, 2022; Mori et al., 2022).

To give a flavour of DatalogMTL as a Knowledge Representation language, we next illustrate its use with an example modeling immunity acquisition against COVID-19 resulting from vaccination or a previous infection.

Example 1. *There is growing evidence that individuals develop COVID-19 immunity for at least 3 months if they got vaccinated and remained without symptoms (or displayed a negative test result) within 3 to 4 weeks following vaccination, or if they were infected within the last 6 months (discounting the last ten days when they had no symptoms) (Feikin et al., 2022). Furthermore, individuals with immunity for at least 5 days display a negative test result. These conditions can be captured by a* DatalogMTL *program* Π_{ex} *with the rules:*

$$\boxplus_{[0,90]} \mathit{Immune}(x) \leftarrow \mathit{NoSympt}(x) \mathcal{S}_{[21,28]} \mathit{Vaccinated}(x), \tag{1}$$

$$\boxplus_{[0,90]} \mathit{Immune}(x) \leftarrow \mathit{NegTest}(x) \wedge \diamondminus_{[21,28]} \mathit{Vaccinated}(x), \tag{2}$$

$$\mathit{Immune}(x) \leftarrow \diamondminus_{(10,183]} \mathit{Infected}(x) \wedge \boxminus_{[0,10]} \mathit{NoSympt}(x), \tag{3}$$

$$\mathit{NegTest}(x) \leftarrow \boxminus_{[0,5]} \mathit{Immune}(x). \tag{4}$$

In particular, Rules (1)–(3) *represent the aforementioned conditions for acquiring 3-month-long immunity. Rule* (1) *checks whether an individual remained without symptoms between* 21 *and* 28 *days* since *receiving vaccination (using the 'since' operator* $\mathcal{S}_{[21,28]}$*); Rule* (2) *checks whether they received a negative test and got vaccinated at some point in the last* 21 *to* 28 *days (using the 'diamond past' operator* $\diamondminus_{[21,28]}$*); in turn, Rule* (3) *checks whether an individual infected at some point in the last six months excluding the last 10 days (operator* $\diamondminus_{(10,183]}$*) remained continuously without symptoms in the last 10 days (using the 'box past' operator* $\boxminus_{[0,10]}$*).*

Assume also that historical data is stored in the form of facts stamped with validity intervals, where the first day of the year is given by the interval $(0, 1]$*, the second day by* $(1, 2]$*, and so on. Ben got vaccinated at 4 p.m. on July 19 (represented as* $199\frac{2}{3}$*). Moreover, Ben had no symptoms since midnight on July 1 (i.e.,* 181*) until noon on August 30 (i.e.,* $242\frac{1}{2}$*). This is represented by a dataset* $\mathcal{D}_{\mathsf{ex}}$ *with the following facts:*

$$\mathit{Vaccinated}(\mathsf{Ben})@199\tfrac{2}{3}, \qquad \mathit{NoSympt}(\mathsf{Ben})@(181, 320\tfrac{1}{2}].$$

DatalogMTL is an expressive language and standard reasoning tasks such as consistency and fact entailment are of high complexity, namely ExpSpace-complete in combined complexity (Brandt et al., 2018) and PSpace-complete in data complexity (Wałęga et al., 2019). Furthermore, worst-case optimal algorithms have comparable best-case and worst-case running times and involve either the construction of large Büchi automata or exponential translations to Linear Temporal Logic (LTL), which makes efficient implementation in data-intensive applications challenging.

In contrast, scalable Datalog implementations often *materialise*—that is, precompute using forward chaining via multiple rounds of rule applications until a fixpoint is reached—all facts entailed by an input program and dataset (Bry et al., 2007; Motik et al., 2014; Carral et al., 2019; Bellomarini et al., 2018). Once a fixpoint has been reached, the facts in the materialisation provide a representation of the canonical, least model of the input over which all queries can be directly answered (Abiteboul et al., 1995; Motik et al., 2019).

As in plain Datalog, each satisfiable pair of a DatalogMTL program and dataset admits also a canonical model defined as the least fixpoint of an immediate consequence operator capturing a single round of rule applications (Brandt et al., 2018; Wałęga et al., 2019). The use of metric temporal operators in rules, however, introduces a number of challenges for

materialisation-based reasoning. In particular, in contrast to Datalog, where materialisation naturally terminates, in DatalogMTL a fixpoint may only be reachable after infinitely many rounds of rule applications.

Example 2. *Consider a program consisting of a rule* $\boxplus_{365} Bday(x) \leftarrow Bday(x)$*, which states that anyone having their birthday at a time point* t *will also be having their birthday at the same time the following year (for simplicity we assume that a year has 365 days). Let us now consider a dataset with a single fact saying that Alan Turing was having his first birthday during the 23rd of June 1913. In the corresponding canonical model, atom* $Bday(\mathsf{Turing})$ *holds at each time within June 23rd of each year from 1913 onwards; the first application of the rule makes* $Bday(\mathsf{Turing})$ *true during the 23rd of June 1914, and each subsequent application makes it true on the same day the year after.*

In our recent work (Wang et al., 2022), we proposed a practical reasoning algorithm for DatalogMTL combining a materialisation-based procedure optimised for efficient rule application with the construction of Büchi automata to ensure completeness and termination. We implemented this approach in the MeTeoR reasoner, which is designed to minimise the use of automata-based techniques in favour of materialisation. In some cases, however, using automata-based techniques was necessary, leading to a performance reduction of orders of magnitude. Thus, identifying fragments of DatalogMTL for which materialisation is guaranteed to terminate can be instrumental for ensuring better scalability and robustness in practice. Furthermore, the fact that materialisation requires infinitely many rounds of rule applications to complete may indicate a modeling error; indeed, the description of an application domain does not typically require unbounded propagation of information along the infinite timeline.

Therefore, in this paper, we propose and study *finitely materialisable* DatalogMTL programs, for which forward chaining is guaranteed to construct a materialisation in a finite number of steps. On the one hand, finitely materialisable programs are naturally well suited for materialisation-based reasoning, which paves the way to the development of efficient implementations; on the other hand, they constitute a natural class of programs sufficiently expressive for many applications of DatalogMTL. Finitely materialisable programs extend both plain Datalog, where programs may be recursive but do not contain metric operators (Abiteboul et al., 1995), and non-recursive DatalogMTL, where the use of metric operators is unrestricted but there are no cyclic dependencies between predicates (Brandt et al., 2018). Hence, finite materialisability can be seen as a safe form of *temporal recursion*. In particular, we will see that the program in Example 1 is finitely materialisable regardless of the dataset despite involving recursion via temporal operators. In contrast, the program from Example 2 is not finitely materialisable, as the repeated application of its immediate consequence operator will infinitely propagate fact $Bday(\mathsf{Turing})$ towards the future.

The main decision problems that we consider in this paper are as follows.

- *Data-dependent finite materialisability*, which is to check whether materialisation of a given DatalogMTL program and dataset will terminate in a finite number of steps,
- *Data-independent finite materialisability*, which is to check if a given DatalogMTL program is finitely materialisable regardless of the dataset, and

– *Fact entailment*, which is to check whether a (finitely materialisable) DatalogMTL program and a given dataset entail a given fact.

Many of our technical results will be applicable to the fragment of DatalogMTL where all intervals in programs and in datasets are *bounded*—that is, where $-\infty$ and ∞ are not mentioned as interval endpoints in either the rules or the data. We refer to such programs and datasets as *bounded*; in particular, the programs and datasets in Examples 1 and 2 are bounded. Finitely materialisable bounded DatalogMTL programs can be used to describe temporal phenomena that have a finite starting and ending point, even if the rules that describe them involve a form of temporal recursion as in the case of Example 1. Thus, such programs are a natural choice for many practical scenarios.

Our contributions in this paper are as follows.

– After recapitulating in Section 2 the relevant background on DatalogMTL, in Section 3 we present the data-dependent and data-independent notions of finite materialisability.

– In Section 4 we study the data-dependent variant of finite materialisability for bounded programs and datasets. We first provide a characterisation of finite materialisability, and then show that each finitely materialisable program and dataset can only entail facts that hold within a specific bounded interval whose length depends on the size of the program and the data. This suggests a materialisation-based decision procedure for checking finite materialisability that works in doubly exponential time. Although this algorithm is well suited for implementation, it is not worst-case optimal, and in Section 4 we provide tight complexity bounds for the problem.

– In Section 5 we study the data-independent variant of finite materialisability for bounded programs. We first show that it reduces to the data-dependent case by considering a single *critical dataset*, which is reminiscent of techniques used for deciding universal termination of variants of the chase procedure for various extensions of Datalog (Gogacz & Marcinkowski, 2014; Cuenca Grau et al., 2013; Marnette & Geerts, 2010), as well as techniques for verifying strong or weak *safety* of temporal programs (Chomicki & Imielinski, 1988; Chomicki, 1990, 1995). This reduction implies that the materialisation algorithm in Section 4 working in doubly exponential time can be adapted to the data-independent setting. We then show that this procedure can be further refined to work in singly exponential time. By establishing a matching lower bound for the problem, we show optimality of our algorithm.

– In Section 6 we propose two incomparable fragments of DatalogMTL, which allow for a limited form of temporal recursion while at the same time ensuring (data-independent) finite materialisability. The fragment of *EDB-guarded* programs requires each rule to contain at least one body atom involving only EDB predicates (i.e., not mentioned in rule heads of the program); in turn, *MTL-acyclic* programs require that (a generalisation of) the program's dependency graph does not contain certain types of cycles.

– In Section 7 we turn our attention to fact entailment. We first consider bounded DatalogMTL programs and datasets and show that reasoning is as hard as for arbitrary DatalogMTL in both combined and data complexity. We then focus on programs

that are finitely materialisable in the data-independent setting and analyse whether fact entailment becomes computationally easier as a result. We show that fact entailment remains PSpace-complete in data complexity, whereas combined complexity drops from ExpSpace-completeness to ExpTime-completeness; furthermore, the same bounds hold already for EDB-guarded programs. Surprisingly, however, fact entailment over MTL-acyclic programs is P-complete in data complexity and ExpTime-complete in combined complexity; thus, it has exactly the same complexity as plain Datalog despite allowing for all types of metric operators in rules and supporting a limited form of temporal recursion. The panorama of complexity and expressivity relations among different fragments of bounded DatalogMTL stemming from the results included in this section is presented in Figure 1.

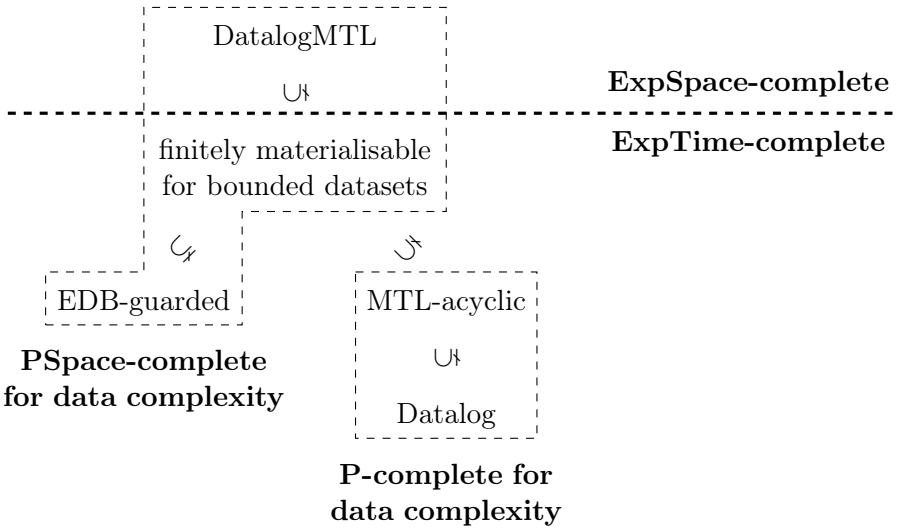


Figure 1: Fragments of bounded DatalogMTL

This paper constitutes a significant extension of our previous conference publication (Wałęga et al., 2021b).

2. Preliminaries

In this section, we recapitulate the syntax and semantics of DatalogMTL (Brandt et al., 2018; Wałęga et al., 2019), introduce the standard reasoning tasks, and describe a generic (possibly non-terminating) materialisation procedure for reasoning over DatalogMTL programs and datasets. We focus on the rational timeline, as opposed to the integer timeline (Wałęga et al., 2020a), and on the standard continuous semantics, as opposed to the alternative pointwise semantics (Ryzhikov et al., 2019).

2.1 Time and Intervals

The *(rational) timeline* is the set $\mathbb{Q}$ of rational numbers; each element of the timeline constitutes a *time point*. We consider binary representations of integers, and represent each

rational number as a fraction with an integer numerator and a positive integer denominator. An *interval* ϱ is a non-empty subset of $\mathbb{Q}$ satisfying the following properties: (i) for all $t_1, t_2, t_3 \in \mathbb{Q}$ with $t_1 < t_2 < t_3$ and $t_1, t_3 \in \varrho$, it is the case that $t_2 \in \varrho$, and (ii) both the greatest lower bound ϱ^- and the least upper bound ϱ^+ of ϱ belong to $\mathbb{Q} \cup \{-\infty, \infty\}$. The bounds ϱ^- and ϱ^+ are called the *left* and *right endpoints* of ϱ, respectively, and $\varrho^+ - \varrho^-$ is the *length* of ϱ. An interval ϱ is *punctual* if it contains exactly one number, it is *positive* if it does not contain negative numbers, and it is *bounded* if both its left and right endpoints are rational numbers. We use the standard representation $\langle \varrho^-, \varrho^+ \rangle$ for interval ϱ, where the *left bracket* $\langle$ is either [or (, the *right bracket* $\rangle$ is either] or), and ϱ^- and ϱ^+ are representations of the left and right endpoints of ϱ, respectively. As usual, brackets [and] indicate that the corresponding endpoints are included in the interval, whereas (and) indicate that they are not included. We abbreviate a punctual interval $[t, t]$ as t. Since different intervals cannot have the same representation, we often abuse notation and identify an interval representation with the interval it represents. Finally, for intervals ϱ and ϱ' we define the operators $\varrho + \varrho' = \{t + t' \mid t \in \varrho \text{ and } t' \in \varrho'\}$, $\varrho - \varrho' = \{t - t' \mid t \in \varrho \text{ and } t' \in \varrho'\}$, and $-\varrho = [0, 0] - \varrho$.

2.2 Syntax of DatalogMTL

We consider a function-free first-order signature. A *relational atom* is a first-order atom of the form $P(\mathbf{s})$, with P an n-ary predicate and $\mathbf{s}$ an n-ary tuple of terms (i.e., constants and variables). A *metric atom* is an expression given by the following grammar, where $P(\mathbf{s})$ ranges over relational atoms and ϱ over positive non-empty intervals:

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

A *rule* is an expression of the form

$$M' \leftarrow M_1 \wedge \cdots \wedge M_n, \qquad \text{for } n \geq 1, \tag{5}$$

where each M_i is a metric atom and M' is generated by the following grammar:[1]

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

The conjunction $M_1 \wedge \cdots \wedge M_n$ in Expression (5) is the rule's *body*, each M_i is a *body atom*, and M' is the rule's *head*. A rule is *safe* if each variable mentioned in the head occurs also in its body, and this occurrence is not in a left operand of $\mathcal{S}$ or $\mathcal{U}$. A rule is *propositional* if all its predicates are nullary and *bounded* if so are all the intervals it mentions and $\top$ does not occur in the body.[2] The *depth* of rule r, written as $\mathsf{depth}(r)$, is the sum of the right endpoints in all intervals occurring in r (or 0 if r mentions no intervals)

A *program* is a finite set of safe rules. A program Π is propositional (resp. bounded) if so are all its rules. The depth of Π, written as $\mathsf{depth}(\Pi)$, is the maximum depth of its rules; hence, bounded programs have finite depth. We let $\mathsf{pred}(\Pi)$ be the number of predicates mentioned in Π, and we let t_Π be the greatest number mentioned in Π (or 1 if Π mentions no numbers). The *dependency graph* of Π is a directed graph with a vertex v_P for each

1. For presentational simplicity, we disallow $\bot$ in rule heads, which ensures consistency and allows us to focus on fact entailment.
2. We disallow $\top$ in rule bodies as a rule $P \leftarrow \top$ simulates a fact $P@(-\infty, \infty)$ over an unbounded interval.

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

Table 1: Semantics of ground metric atoms

predicate P in Π and an edge (v_Q, v_R) if there is a rule mentioning Q in its body and R in its head. A program is *recursive* if its dependency graph is cyclic.

An expression (metric atom, rule, or program) is *ground* if it mentions no variables. A *metric fact* over an interval ϱ is an expression $M@\varrho$, with M a ground metric atom; it is relational if so is M and bounded if so is ϱ. A *dataset* is a finite set of relational facts; it is bounded if so is each of its facts. For a dataset $\mathcal{D}$, we let $t_{\mathcal{D}}^{\min}$ and $t_{\mathcal{D}}^{\max}$ be the minimal and maximal numbers mentioned as interval endpoints in $\mathcal{D}$ (if $\mathcal{D}$ does not mention any numbers, we let both $t_{\mathcal{D}}^{\min}$ and $t_{\mathcal{D}}^{\max}$ be 0).

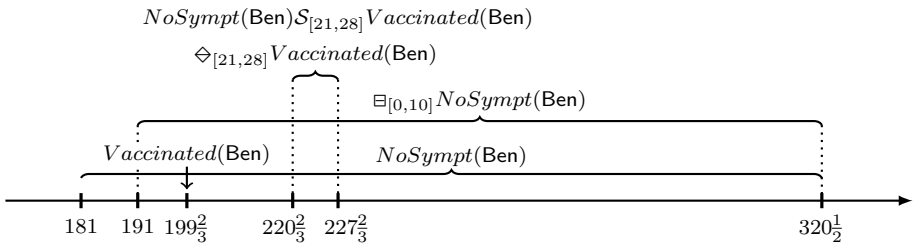
The *grounding* of a program Π, written as $\mathsf{ground}(\Pi)$, is the set of ground rules obtained by assigning constants in the signature to variables in Π. The *grounding* of Π with respect to a dataset $\mathcal{D}$, written as $\mathsf{ground}(\Pi, \mathcal{D})$, is the restriction of $\mathsf{ground}(\Pi)$ to ground rules containing only constants from Π and $\mathcal{D}$.

2.3 Semantics of DatalogMTL

An *interpretation* $\mathfrak{I}$ is a function which specifies, for each ground relational atom $P(\mathbf{s})$ and each time point t, whether $P(\mathbf{s})$ is *satisfied* at t, in which case we write $\mathfrak{I}, t \models P(\mathbf{s})$. This notion extends to other ground metric atoms as given in Table 1.

Example 3. *Consider an interpretation $\mathfrak{I}$ represented by the dataset $\mathcal{D}_{\mathsf{ex}}$ from Example 1, that is, $\mathfrak{I}, 199\frac{2}{3} \models$ Vaccinated(Ben) and $\mathfrak{I}, t \models$ NoSympt(Ben), for all $t \in (181, 320\frac{1}{2}]$. By the semantics of metric operators in Table 1, the following hold (as depicted in the figure below):*

- $\mathfrak{I}, t \models \diamondminus_{[21,28]}$ *Vaccinated*(Ben) *for all $t \in [220\frac{2}{3}, 227\frac{2}{3}]$, since for each of these time points t, Vaccinated*(Ben) *holds in some time point within the interval $[t - 28, t - 21]$.*
- $\mathfrak{I}, t \models \boxminus_{[0,10]}$ *NoSympt*(Ben) *for all $t \in (191, 320\frac{1}{2}]$, as for any such t, NoSympt*(Ben) *holds continuously in the interval $[t - 10, t - 0]$.*
- $\mathfrak{I}, t \models$ *NoSympt*(Ben)$\mathcal{S}_{[21,28]}$ *Vaccinated*(Ben) *for all $t \in [220\frac{2}{3}, 227\frac{2}{3}]$, as for all these t, Vaccinated*(Ben) *holds in some time point in $[t - 28, t - 21]$ such that NoSympt*(Ben) *holds continuously between this time point and t.*



An interpretation $\mathfrak{I}$ *satisfies a metric fact* $M@\varrho$, written $\mathfrak{I} \models M@\varrho$, if $\mathfrak{I}, t \models M$ for all $t \in \varrho$. Moreover, $\mathfrak{I}$ *satisfies a ground rule* r if, whenever $\mathfrak{I}$ satisfies each body atom of r at a time point t, then $\mathfrak{I}$ also satisfies the head of r at t. Furthermore, $\mathfrak{I}$ *satisfies a rule* r if it satisfies each rule in $\mathsf{ground}(\{r\})$. We say that $\mathfrak{I}$ is a *model of a program* Π if $\mathfrak{I}$ satisfies each rule in Π. An interpretation $\mathfrak{I}$ is a *model of a set of metric facts* if it satisfies each of these facts. A set $\mathcal{M}$ of metric facts *entails* a metric fact $M@\varrho$ if each model of $\mathcal{M}$ is also a model of $M@\varrho$. A program Π and a set $\mathcal{M}$ of metric facts *entail* a set $\mathcal{M}'$ of metric facts, written $(\Pi, \mathcal{M}) \models \mathcal{M}'$, if each model of both Π and $\mathcal{M}$ is also a model of $\mathcal{M}'$; we will often write $\mathcal{M} \models \mathcal{M}'$ instead of $(\emptyset, \mathcal{M}) \models \mathcal{M}'$. If $\mathcal{M}$ or $\mathcal{M}'$ is a singleton, say $\{M@\varrho\}$, then we may omit curly brackets and write $M@\varrho \models \mathcal{M}'$ and $\mathcal{M} \models M@\varrho$, respectively.

An interpretation $\mathfrak{I}$ *contains* an interpretation $\mathfrak{I}'$, written $\mathfrak{I}' \subseteq \mathfrak{I}$, if $\mathfrak{I}', t \models P(\mathbf{s})$ implies $\mathfrak{I}, t \models P(\mathbf{s})$, for each ground relational atom $P(\mathbf{s})$ and each time point t. We say that $\mathfrak{I}$ is the *least interpretation* in a set X of interpretations if $\mathfrak{I} \subseteq \mathfrak{I}'$, for every $\mathfrak{I}' \in X$. Each dataset $\mathcal{D}$ admits the least interpretation $\mathfrak{I}_\mathcal{D}$ among all models of $\mathcal{D}$; we say that a dataset $\mathcal{D}$ *represents* an interpretation $\mathfrak{I}$ if $\mathfrak{I} = \mathfrak{I}_\mathcal{D}$. Furthermore, for an interpretation $\mathfrak{I}$ and an interval ϱ, we let the *projection* of $\mathfrak{I}$ over ϱ, written $\mathfrak{I}|_\varrho$, be the interpretation that coincides with $\mathfrak{I}$ over ϱ and does not satisfy any relational atoms outside ϱ.

The *immediate consequence operator* T_Π, for a program Π, is a function mapping an interpretation $\mathfrak{I}$ to the least interpretation containing $\mathfrak{I}$ and satisfying the following property for each $r \in \mathsf{ground}(\Pi)$: whenever $\mathfrak{I}$ satisfies each body atom of r at a time point t, then $T_\Pi(\mathfrak{I})$ satisfies the head of r at t. The successive application of T_Π to $\mathfrak{I}_\mathcal{D}$ defines a transfinite sequence of interpretations $T_\Pi^\alpha(\mathfrak{I}_\mathcal{D})$, for ordinals α, as follows:

$$
\begin{aligned}
&T_\Pi^0(\mathfrak{I}_\mathcal{D}) = \mathfrak{I}_\mathcal{D}, \\
&T_\Pi^\alpha(\mathfrak{I}_\mathcal{D}) = T_\Pi\left(T_\Pi^{\alpha-1}(\mathfrak{I}_\mathcal{D})\right), && \text{for } \alpha \text{ a successor ordinal,} \\
&T_\Pi^\alpha(\mathfrak{I}_\mathcal{D}) = \bigcup_{\beta<\alpha} T_\Pi^\beta(\mathfrak{I}_\mathcal{D}), && \text{for } \alpha \text{ a limit ordinal.}
\end{aligned}
$$

The *canonical interpretation* $\mathfrak{C}_{\Pi,\mathcal{D}}$ of Π and $\mathcal{D}$ is the interpretation $T_\Pi^{\omega_1}(\mathfrak{I}_\mathcal{D})$, where ω_1 is the first uncountable ordinal. Since we do not allow $\bot$ in rule heads, $\mathfrak{C}_{\Pi,\mathcal{D}}$ is the least model of Π and $\mathcal{D}$ (Brandt et al., 2017).

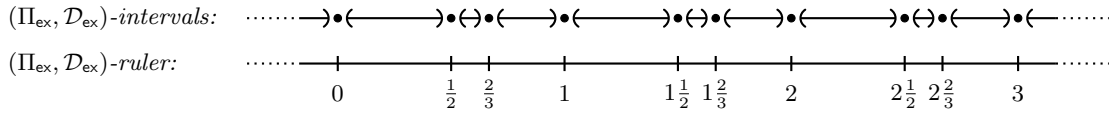
Example 4. *Consider the program* Π_ex *and dataset* $\mathcal{D}_\mathsf{ex}$ *from Example 1. The interpretations* $T_{\Pi_\mathsf{ex}}^\alpha(\mathfrak{I}_{\mathcal{D}_\mathsf{ex}})$ *are represented by the following datasets:*

$T^0_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$:	$\mathit{Vaccinated}(\mathsf{Ben})@199\frac{2}{3}$, $\mathit{NoSympt}(\mathsf{Ben})@(181, 320\frac{1}{2}]$
$T^1_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$:	$\mathit{Vaccinated}(\mathsf{Ben})@199\frac{2}{3}$, $\mathit{NoSympt}(\mathsf{Ben})@(181, 320\frac{1}{2}]$, $\mathit{Immune}(\mathsf{Ben})@[220\frac{2}{3}, 317\frac{2}{3}]$
$T^2_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$:	$\mathit{Vaccinated}(\mathsf{Ben})@199\frac{2}{3}$, $\mathit{NoSympt}(\mathsf{Ben})@(181, 320\frac{1}{2}]$, $\mathit{Immune}(\mathsf{Ben})@[220\frac{2}{3}, 317\frac{2}{3}]$, $\mathit{NegTest}(\mathsf{Ben})@[225\frac{2}{3}, 317\frac{2}{3}]$
$T^3_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$:	same facts as in $T^2_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$

Since $T^2_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}}) = T^3_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$*, we obtain that* $T^2_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}}) = T^\alpha_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$*, for any ordinal* $\alpha \geq 2$*, and so,* $T^2_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}}) = \mathfrak{C}_{\Pi_{\mathsf{ex}},\mathcal{D}_{\mathsf{ex}}}$.

The canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$ is regular (Wałęga et al., 2019), as described next. We let the $(\Pi, \mathcal{D})$-*ruler* be the set of all time points of the form $t + i \cdot \mathsf{div}(\Pi)$, for t a time point mentioned in $\mathcal{D}$ (as an endpoint of some interval), $i \in \mathbb{Z}$, and $\mathsf{div}(\Pi) = \frac{1}{k}$, where k is the product of all denominators occurring in the representation of the rational endpoints of the intervals mentioned in Π (if Π has no intervals with rational endpoints, then $k = 1$, and hence, $\mathsf{div}(\Pi) = 1$). A $(\Pi, \mathcal{D})$-*interval* is either a punctual interval over a time point on the $(\Pi, \mathcal{D})$-ruler, or an interval of the form (t_1, t_2), where t_1 and t_2 are consecutive time points on the $(\Pi, \mathcal{D})$-ruler. Then, we say than an interpretation $\mathfrak{I}$ is a $(\Pi, \mathcal{D})$-*interpretation* if, for every relational fact $M@t$, it holds that $\mathfrak{I} \models M@t$ implies $\mathfrak{I} \models M@\varrho$, where ϱ is the $(\Pi, \mathcal{D})$-interval containing t. The canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$, as well the interpretations $T^\alpha_\Pi(\mathfrak{I}_\mathcal{D})$ for any ordinal α, are $(\Pi, \mathcal{D})$-interpretations (Wałęga et al., 2019).

Example 5. *Consider program* Π_{ex} *and the dataset* $\mathcal{D}_{\mathsf{ex}}$ *from Example 1. Since all rational numbers occurring in* Π_{ex} *are integers,* $\mathsf{div}(\Pi_{\mathsf{ex}}) = 1$*. Furthermore, as* 181*,* $199\frac{2}{3}$*, and* $320\frac{1}{2}$ *are the only numbers occurring in* $\mathcal{D}_{\mathsf{ex}}$*, the* $(\Pi_{\mathsf{ex}}, \mathcal{D}_{\mathsf{ex}})$*-ruler consists of all rational numbers of the form* i*,* $\frac{1}{2}+i$*, and* $\frac{2}{3}+i$*, for any integer* i*. Hence,* $(\Pi_{\mathsf{ex}}, \mathcal{D}_{\mathsf{ex}})$*-intervals are, for example,* $[0, 0]$*,* $(0, \frac{1}{2})$*,* $[\frac{1}{2}, \frac{1}{2}]$*, and* $(\frac{1}{2}, \frac{2}{3})$*, as depicted below.*



2.4 Reasoning Problems and Complexity of Reasoning

In this paper, we focus on *fact entailment*, which is the problem of checking whether a relational fact is entailed by a program and a dataset. We consider both *combined* and *data* complexity of this problem. In combined complexity, we treat as inputs all among the dataset, program, and query fact. In data complexity, only the dataset is considered as input, whereas the program and query fact are considered fixed.

Fact entailment in DatalogMTL is of high complexity; it is ExpSpace-complete (Brandt et al., 2018) and PSpace-complete in data complexity (Wałęga et al., 2019). Furthermore, PSpace-hardness in data complexity holds for the *core* fragment (Wałęga et al., 2020b), where rules are restricted to contain a single atom in the body. Lower complexity fragments have also been identified. In particular, for non-recursive programs fact entailment is PSpace-complete in combined complexity and in AC^0 for data complexity (Brandt et al., 2017); tractability in data complexity can also be achieved by disallowing certain metric operators in linear and core fragments (Wałęga et al., 2020b).

Algorithm 1: ApplyRules

Input: A program Π and a dataset $\mathcal{D}$

Output: A dataset representing $T_\Pi(\mathfrak{I}_\mathcal{D})$

1 $\mathcal{N} \coloneqq \emptyset$; // initialise set $\mathcal{N}$ of newly derived facts

2 **for** *each rule* $r \in \mathsf{ground}(\Pi, \mathcal{D})$ **do**

3 $M_1, \dots, M_n \coloneqq$ the body atoms of r;

4 $M' \coloneqq$ the head of r;

5 $P(\mathbf{c}) \coloneqq$ the relational atom in M';

6 **for** *each* $i \in \{1, \dots, n\}$ **do** $\mathsf{I}_{M_i} \coloneqq \big\{ \subseteq\text{-maximal } \varrho \mid \mathcal{D} \models M_i @ \varrho \big\}$;

7 $\mathsf{I}_{M'} \coloneqq \big\{ \subseteq\text{-maximal } \varrho \mid \varrho \subseteq (\varrho_1 \cap \dots \cap \varrho_n), \text{ for some } \varrho_1 \in \mathsf{I}_{M_1}, \dots, \varrho_n \in \mathsf{I}_{M_n} \big\}$;

8 $\mathsf{I}_{P(\mathbf{c})} \coloneqq \big\{ \subseteq\text{-maximal } \varrho \mid M' @ \varrho' \models P(\mathbf{c}) @ \varrho, \text{ for some } \varrho' \in \mathsf{I}_{M'} \big\}$;

9 $\mathcal{N} \coloneqq \mathcal{N} \cup \big\{ P(\mathbf{c}) @ \varrho \mid \varrho \in \mathsf{I}_{P(\mathbf{c})} \big\}$; // update the set $\mathcal{N}$

10 **return** $\mathcal{D} \cup \mathcal{N}$;

2.5 Materialisation-based Reasoning

Materialisation-based reasoning is the most common technique of choice implemented in scalable Datalog reasoners (Bry et al., 2007; Motik et al., 2014; Carral et al., 2019; Bellomarini et al., 2018). Facts entailed by a program and a dataset are derived in successive rounds of rule applications until a fixpoint is reached and a *full materialisation* has been computed. At this point, queries can be answered directly over the computed materialisation and rules do not need to be considered any further.

In this section, we formulate a (possibly non-terminating) generic reasoning procedure for DatalogMTL based on materialisation. An instantiation of this procedure has been implemented, for instance, in the MeTeoR system (Wang et al., 2022).

Rules are applied by first identifying maximal intervals in which ground body atoms hold simultaneously, and then determining maximal intervals in which the head atoms can be derived. Details are presented in Algorithm 1, which performs a single round of rule applications for a given input program and dataset.

Example 6. *Given the program* Π_{ex} *and dataset* $\mathcal{D}_{\mathsf{ex}}$ *from Example 1, Algorithm 1 applies Rule* (1) *by first identifying* $\varrho = [220\frac{2}{3}, 227\frac{2}{3}]$ *as the maximal interval such that we have* $\mathcal{D}_{\mathsf{ex}} \models \mathit{NoSympt}(x)\mathcal{S}_{[21,28]}\mathit{Vaccinated}(x)@\varrho$. *Hence, the application of the rule derives* $\mathit{Immune}(\mathsf{Ben})@[220\frac{2}{3}, 317\frac{2}{3}]$, *as shown in the table from Example 4.*

The result of applying Algorithm 1 to Π and $\mathcal{D}$ is always a dataset (i.e., a finite object), and the following proposition establishes that this dataset represents the infinite interpretation obtained by a single application of the immediate consequence operator associated to Π to the dataset $\mathcal{D}$. Thus, Algorithm 1 provides a syntactic counterpart to the application of the immediate consequence operator.

Proposition 7. *On input* Π *and* $\mathcal{D}$, *Algorithm 1 outputs a dataset representing* $T_\Pi(\mathfrak{I}_\mathcal{D})$.

Proof. In each iteration of the loop, Algorithm 1 extends (in Line 9) the dataset $\mathcal{N}$ with $\big\{ P(\mathbf{c})@\varrho \mid \varrho \in \mathsf{I}_{P(\mathbf{c})} \big\}$. Thus, to prove that the output $\mathcal{D} \cup \mathcal{N}$ of Algorithm 1 represents

Procedure 2: Materialisation-based reasoning

Input: A program Π and a dataset $\mathcal{D}$
Output: A dataset representing $\mathfrak{C}_{\Pi,\mathcal{D}}$

1 $\mathcal{D}_{\mathsf{new}} \coloneqq \mathcal{D}$; // initialise $\mathcal{D}_{\mathsf{new}}$
2 **repeat**
3 $\quad \mathcal{D}_{\mathsf{old}} \coloneqq \mathcal{D}_{\mathsf{new}}$; // copy $\mathcal{D}_{\mathsf{new}}$ before applying rules
4 $\quad \mathcal{D}_{\mathsf{new}} \coloneqq \mathsf{ApplyRules}(\Pi, \mathcal{D}_{\mathsf{new}})$;
5 **until** $\mathfrak{I}_{\mathcal{D}_{\mathsf{old}}} = \mathfrak{I}_{\mathcal{D}_{\mathsf{new}}}$;
6 **return** $\mathcal{D}_{\mathsf{new}}$;

$T_\Pi(\mathfrak{I}_\mathcal{D})$, it suffices to show that, after performing one iteration of the loop (Lines 3–9) for a rule $r \in \mathsf{ground}(\Pi, \mathcal{D})$, the dataset $\mathcal{D} \cup \{P(\mathbf{c})@\varrho \mid \varrho \in \mathsf{I}_{P(\mathbf{c})}\}$ represents $T_{\{r\}}(\mathfrak{I}_\mathcal{D})$.

Indeed, for each body atom M_i of r, set I_{M_i} (computed in Line 6) consists of intervals containing exactly those time points for which M_i holds in $\mathfrak{I}_\mathcal{D}$. Thus, $\mathsf{I}_{M'}$ (computed in Line 7) consists of intervals containing exactly those time points in which all the body atoms of r simultaneously hold in $\mathfrak{I}_\mathcal{D}$. Then intervals in $\mathsf{I}_{P(\mathbf{c})}$ (computed in Line 8) contain exactly those time points for which the head atom $P(\mathbf{c})$ of r is entailed by M' holding in intervals from $\mathsf{I}_{M'}$. Therefore, $\mathcal{D} \cup \{P(\mathbf{c})@\varrho \mid \varrho \in \mathsf{I}_{P(\mathbf{c})}\}$ represents $T_{\{r\}}(\mathfrak{I}_\mathcal{D})$, as required. □

We next introduce Procedure 2 which, given a program Π and a dataset $\mathcal{D}$ as input, performs materialisation by iteratively calling Algorithm 1 to compute a sequence $\mathcal{D}_0, \mathcal{D}_1, \dots$ of datasets, where each $\mathcal{D}_i$ represents the interpretation $T^i_\Pi(\mathfrak{I}_\mathcal{D})$. Indeed, the following Proposition 8 is a direct consequence of Proposition 7.

Proposition 8. *After $k \in \mathbb{N}$ iterations of the loop from Algorithm 2 on input Π and $\mathcal{D}$, the dataset $\mathcal{D}_{\mathsf{new}}$ represents $T^k_\Pi(\mathfrak{I}_\mathcal{D})$.*

If $\mathcal{D}_i = \mathcal{D}_{i+1}$ at some point in the sequence $\mathcal{D}_0, \mathcal{D}_1, \dots$ computed by Procedure 2, then Procedure 2 outputs $\mathcal{D}_i$. In particular, if we consider program Π_{ex} and dataset $\mathcal{D}_{\mathsf{ex}}$ from Example 1 as input, Procedure 2 will stop after three iterations and return the dataset representing $T^2_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$, as shown in Example 4.

In general, however, the procedure may not terminate. Indeed, for some programs and datasets, the construction of the canonical interpretation requires infinitely many applications of the immediate consequence operator.

Example 9. *Consider a program $\Pi = \{\boxplus_{[0,1]} P \leftarrow P\}$ and a dataset $\mathcal{D} = \{P@[0,1]\}$. If the proposition P holds at some time point t, then Π ensures that P holds also in the interval $[t, t+1]$. It follows that $T^i_\Pi(\mathfrak{I}_\mathcal{D}) \models P@[i, i+1]$, for each $i \in \mathbb{N}$, and that the canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$ is reached only after ω applications of T_Π to $\mathfrak{I}_\mathcal{D}$.*

The main research question investigated in this paper is to determine for which programs and datasets application of the immediate consequence operator reaches a fixpoint in finitely many steps; that is, when Procedure 2 is guaranteed to terminate. In such cases, the procedure computes a full materialisation and becomes a decision procedure that can be used for checking entailment of arbitrary facts.

3. Finitely Materialisable DatalogMTL Programs

In this section, we introduce the notion of *finitely materialisable* DatalogMTL *programs*. Although our definition is based on the semantics of the immediate consequence operator and the canonical interpretation, Proposition 8 ensures that it admits an equivalent formulation based on the termination of the generic materialisation procedure in Section 2.5.

Definition 10. *Let* Π *be a program and* α *an ordinal number. We say that the immediate consequence operator* T_Π converges for a dataset $\mathcal{D}$ in α steps *if* $T_\Pi^\alpha(\mathfrak{I}_\mathcal{D}) = \mathfrak{C}_{\Pi,\mathcal{D}}$. *Program* Π *is* finitely materialisable for a dataset $\mathcal{D}$ *if* T_Π *converges for* $\mathcal{D}$ *in some finite number of steps (or, equivalently, if Procedure 2 terminates on input* Π *and* $\mathcal{D}$*).*

It follows from Proposition 8 that, for each program Π that is finitely materialisable for a dataset $\mathcal{D}$, there exists a dataset representing the canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$. The converse, however, does not hold: the canonical interpretation of a program Π and a dataset $\mathcal{D}$ may admit a finite representation, but Π may be not finitely materialisable for $\mathcal{D}$.

Example 11. *As shown in Example 4, program* Π_{ex} *is finitely materialisable for dataset* $\mathcal{D}_{\mathsf{ex}}$*, and the dataset representing* $T^2_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$ *also represents the canonical interpretation. In contrast, if we consider program* $\Pi = \{\boxplus_{[0,1]} P \leftarrow P\}$ *and dataset* $\mathcal{D} = \{P@[0,1]\}$ *from Example 9, we can observe that the canonical interpretation can be represented by a single fact* $P@[0,\infty)$*, whereas the operator* T_Π *does not converge for* $\mathcal{D}$ *in a finite number of steps.*

The notion of finite materialisability in Definition 10 is data dependent, in that a given program could be finitely materialisable for a given dataset, but perhaps not for a different dataset. We next provide a data-independent notion of finite materialisability which is parametrised by a class of datasets, where natural instantiations include the classes of all datasets or all bounded datasets.

Definition 12. *A program* Π *is* finitely materialisable for a class *of datasets if* Π *is finitely materialisable for each dataset in this class.*

A natural example of DatalogMTL programs that are guaranteed to be finitely materialisable for every dataset are non-recursive programs. Indeed, divergence of the immediate consequence operator is always due to the presence of recursion involving metric atoms. This follows immediately from the following proposition, which also establishes a bound on the number of applications of the immediate consequence operator required to compute the canonical interpretation.

Proposition 13. *For any non-recursive program* Π *and for any dataset* $\mathcal{D}$*, it holds that* $T_\Pi^{\mathsf{pred}(\Pi)-1}(\mathfrak{I}_\mathcal{D}) = \mathfrak{C}_{\Pi,\mathcal{D}}$.

Proof. For each predicate P in Π or $\mathcal{D}$, we let $d(P)$ be the length of a longest path in the dependency graph of Π, which ends in v_P (or 0 if no such path exists). Note that since Π is non-recursive, its dependency graph has no cycles, and so, $d(P) \leq \mathsf{pred}(\Pi) - 1$, for each P. Thus, it suffices to show that, for each relational fact $P(\mathbf{c})@t$, if $\mathfrak{C}_{\Pi,\mathcal{D}} \models P(\mathbf{c})@t$, then $T_\Pi^{d(P)}(\mathfrak{I}_\mathcal{D}) \models P(\mathbf{c})@t$. We proceed by induction on $d(P)$.

For the base case, assume that $\mathfrak{C}_{\Pi,\mathcal{D}} \models P(\mathbf{c})@t$, for some P with $d(P) = 0$. Since $d(P) = 0$, either P does not occur in Π, or P occurs in Π and v_P has no incoming edges. In both cases $\mathfrak{C}_{\Pi,\mathcal{D}} \models P(\mathbf{c})@t$ implies that $\mathfrak{I}_{\mathcal{D}} \models P(\mathbf{c})@t$, and so, $T^0_\Pi(\mathfrak{I}_{\mathcal{D}}) \models P(\mathbf{c})@t$.

For the inductive step, assume that $\mathfrak{C}_{\Pi,\mathcal{D}} \models P(\mathbf{c})@t$, where $d(P) > 0$. If $\mathfrak{I}_{\mathcal{D}} \models P(\mathbf{c})@t$, then $T^0_\Pi(\mathfrak{I}_{\mathcal{D}}) \models P(\mathbf{c})@t$, and so, $T^{d(P)}_\Pi(\mathfrak{I}_{\mathcal{D}}) \models P(\mathbf{c})@t$. Otherwise, there exists a rule r in $\mathsf{ground}(\Pi, \mathcal{D})$ and a time point t' such that all the body atoms of r are satisfied at t' and the head of r being satisfied at t' entails $P(\mathbf{c})@t$. By the definition of the dependency graph, we have $d(Q) < d(P)$ for each predicate Q occurring in the body of r. Therefore, by the inductive assumption, all the body atoms of r must be satisfied at t' in $T^{d(P)-1}_\Pi(\mathfrak{I}_{\mathcal{D}})$. Consequently, $T_{\{r\}}\left(T^{d(P)-1}_\Pi(\mathfrak{I}_{\mathcal{D}})\right) \models P(\mathbf{c})@t$, and so, $T^{d(P)}_\Pi(\mathfrak{I}_{\mathcal{D}}) \models P(\mathbf{c})@t$. □

Moreover, the following example shows that the bound on the number of applications of the immediate consequence operator in Proposition 13 is optimal.

Example 14. *Consider an arbitrary $n \in \mathbb{N}$ and a program Π_n comprising the rules*

$$\boxplus_1 P_1 \leftarrow P_0, \qquad \boxplus_1 P_2 \leftarrow P_1, \qquad \boxplus_1 P_3 \leftarrow P_2, \qquad \ldots, \qquad \boxplus_1 P_n \leftarrow P_{n-1}.$$

Clearly, Π_n is non-recursive and $\mathsf{pred}(\Pi) - 1 = n$. Now, consider a dataset $\mathcal{D} = \{P_0@0\}$. We obtain that, for each $i \leq n$, the interpretation $T^i_{\Pi_n}(\mathfrak{I}_{\mathcal{D}})$ entails $\{P_j@j \mid j \leq i\}$. Thus, $T^n_{\Pi_n}(\mathfrak{I}_{\mathcal{D}}) = \mathfrak{C}_{\Pi_n,\mathcal{D}}$, so the bound on the number of rounds of rule application to materialise a non-recursive program, established in Proposition 13, cannot be decreased.

The presence of recursion via metric atoms is, however, not always harmful, and many recursive programs are finitely materialisable. For instance, program Π_{ex} from Example 1 is recursive, as its dependency graph has a cycle induced by Rules (2) and (4); however, $T_{\Pi_{\mathsf{ex}}}$ converges in at most two iterations for any dataset.

In the remainder of this paper, we study the main decision problems in our setting. In Section 4, we study finite materialisability of a bounded program with respect to a single bounded dataset; furthermore, in Section 5, we extend our study to consider the problem of checking whether a given bounded program is finitely materialisable for the class of all bounded datasets. Finally, in Section 7.2 we study the complexity of checking fact entailment over finitely materialisable bounded programs and datasets.

4. Data-dependent Finite Materialisability

In this section, we study the problem of deciding whether a given bounded program Π is finitely materialisable for a given bounded dataset $\mathcal{D}$.

In Section 4.1, we provide a characterisation of finite materialisability in terms of the number of facts (over $(\Pi, \mathcal{D})$-intervals) entailed by Π and $\mathcal{D}$, which allows us to focus our attention on analysing the kinds of temporal facts that can be entailed by a finitely materialisable program and dataset. In Section 4.2 we show that, if Π is finitely materialisable for $\mathcal{D}$, then all facts entailed by Π and $\mathcal{D}$ hold within a specific bounded interval ϱ, and we provide a bound on the length of ϱ in terms of the size of Π and $\mathcal{D}$. This result suggests a materialisation-based decision procedure for checking finite materialisability, where materialisation is performed until a fixpoint is reached (in which case Π is finitely materialisable

for $\mathcal{D}$) or until we derive a fact with an interval not contained within ϱ (in which case, Π is not finitely materialisable for $\mathcal{D}$). We formulate such algorithm in Section 4.3 and show that it works in doubly exponential time in both the sizes of Π and $\mathcal{D}$. This algorithm, though well suited for implementation, is not worst-case optimal and in Section 4.4 we establish tight combined and data complexity bounds for our problem.

4.1 Characterising Data-dependent Finite Materialisability

We now provide our characterisation of data-dependent finite materialisability. To this end, we start by showing that rule application in DatalogMTL is temporally localised; in particular, a relational fact at a time point t can only be derived in a single round of rule applications from facts that hold within the interval $[t - \mathsf{depth}(\Pi), t + \mathsf{depth}(\Pi)]$, where we recall that bounded programs are of finite depth (c.f. Section 2.2) and hence the application of a rule cannot derive a fact that is 'too far away' on the timeline.

Lemma 15. *Assume that $T_\Pi(\mathfrak{I}) \models M@t$, for a program Π, an interpretation $\mathfrak{I}$, and a relational fact $M@t$. Then $T_\Pi(\mathfrak{I}') \models M@t$, where $\mathfrak{I}'$ is the projection of $\mathfrak{I}$ over the interval $[t - \mathsf{depth}(\Pi), t + \mathsf{depth}(\Pi)]$.*

Proof. Assume that $T_\Pi(\mathfrak{I}) \models M@t$. We will consider separately the cases when $\mathfrak{I} \models M@t$ and $\mathfrak{I} \not\models M@t$. First, assume that $\mathfrak{I} \models M@t$. Hence $\mathfrak{I}', t \models M@t$, due to the fact that $t \in [t - \mathsf{depth}(\Pi), t + \mathsf{depth}(\Pi)]$ and $\mathfrak{I}'$ is projection of $\mathfrak{I}$ over $[t - \mathsf{depth}(\Pi), t + \mathsf{depth}(\Pi)]$.

Second, assume that $\mathfrak{I} \not\models M@t$. Since $T_\Pi(\mathfrak{I}) \models M@t$, there exist a rule $r \in \mathsf{ground}(\Pi, \mathcal{D})$ and a time point t' such that the application of r to $\mathfrak{I}$ at t' derives $M@t$; that is, all body atoms of r are satisfied in $\mathfrak{I}$ at t' and $M'@t' \models M@t$, where M' is the head of r. Let $\mathcal{M}$ be any subset-minimal set of relational facts over punctual intervals which guarantees that all the body atoms of r are satisfied at t'. Moreover, let d_B and d_H be the sums of all right endpoints of intervals mentioned in the body and in the head of r, respectively. Then, by the semantics of metric temporal operators, all the facts in $\mathcal{M}$ are over time points located no further than d_B from t'. Similarly, the distance between t' and t cannot exceed d_H. Hence, all the facts in $\mathcal{M}$ are over time points located no further than $d_B + d_H$ from t. Now, by the definition of depth, we obtain that $d_B + d_H \leq \mathsf{depth}(\Pi)$, so $\mathfrak{I}'$ must satisfy all facts in $\mathcal{M}$. This means that $T_{\{r\}}(\mathfrak{I}') \models M@t$, and thus, $T_\Pi(\mathfrak{I}') \models M@t$. □

We can now exploit Lemma 15 to establish our characterisation of finite materialisability.

Theorem 16. *A bounded program Π is finitely materialisable for a bounded dataset $\mathcal{D}$ if and only if Π and $\mathcal{D}$ entail finitely many relational facts over $(\Pi, \mathcal{D})$-intervals.*

Proof. For the forward direction, assume that Π is finitely materialisable for $\mathcal{D}$, so there exists $k \in \mathbb{N}$ such that $T^k_\Pi(\mathfrak{I}_\mathcal{D}) = \mathfrak{C}_{\Pi,\mathcal{D}}$. We show inductively on $i \in \mathbb{N}$ that each interpretation $T^i_\Pi(\mathfrak{I}_\mathcal{D})$ entails finitely many relational facts over $(\Pi, \mathcal{D})$-intervals. This will imply that $\mathfrak{C}_{\Pi,\mathcal{D}}$ also entails finitely many relational facts over $(\Pi, \mathcal{D})$-intervals.

In the base case, we have $T^0_\Pi(\mathfrak{I}_\mathcal{D}) = \mathfrak{I}_\mathcal{D}$ and $\mathfrak{I}_\mathcal{D}$ entails finitely many relational facts over $(\Pi, \mathcal{D})$-intervals since $\mathcal{D}$ is bounded. For the inductive step, assume that $T^i_\Pi(\mathfrak{I}_\mathcal{D})$ entails finitely many relational facts over $(\Pi, \mathcal{D})$-intervals. Hence, there exists a bounded interval ϱ such that all the relational facts entailed by $T^i_\Pi(\mathfrak{I}_\mathcal{D})$ are over intervals contained in ϱ. Since each bounded interval contains a finite number of $(\Pi, \mathcal{D})$-intervals, it suffices to show that

all the relational facts entailed by $T_\Pi^{i+1}(\mathfrak{I}_\mathcal{D})$ lie within a bounded interval. In particular, we show that they lie within $[\varrho^- - \mathsf{depth}(\Pi), \varrho^+ + \mathsf{depth}(\Pi)]$. To this end, assume that $T_\Pi^{i+1}(\mathfrak{I}_\mathcal{D})$ entails a relational fact $M@t$. Hence, by Lemma 15, we obtain that $T_\Pi(\mathfrak{I}') \models M@t$, where $\mathfrak{I}'$ is the projection of $T_\Pi^i(\mathfrak{I}_\mathcal{D})$ over $[t - \mathsf{depth}(\Pi), t + \mathsf{depth}(\Pi)]$. Next, we show that $[t - \mathsf{depth}(\Pi), t + \mathsf{depth}(\Pi)]$ must overlap with ϱ. Indeed, if this was not the case, then $\mathfrak{I}'$ would be an empty interpretation, so $T_\Pi(\mathfrak{I}')$ would not entail $M@t$ (nor any other relational fact), which contradicts our assumption. Hence, $t + \mathsf{depth}(\Pi) \geq \varrho^-$ and $t - \mathsf{depth}(\Pi) \leq \varrho^+$. Thus, $t \geq \varrho^- - \mathsf{depth}(\Pi)$ and $t \leq \varrho^+ + \mathsf{depth}(\Pi)$, that is, $t \in [\varrho^- - \mathsf{depth}(\Pi), \varrho^+ + \mathsf{depth}(\Pi)]$.

For the converse direction, assume that Π and $\mathcal{D}$ entail $k \in \mathbb{N}$ relational facts over $(\Pi, \mathcal{D})$-intervals. We show that $T_\Pi^{k+1}(\mathfrak{I}_\mathcal{D}) = T_\Pi^k(\mathfrak{I}_\mathcal{D})$. Assume, towards a contradiction, that $T_\Pi^{k+1}(\mathfrak{I}_\mathcal{D}) \neq T_\Pi^k(\mathfrak{I}_\mathcal{D})$. Hence $T_\Pi^{i+1}(\mathfrak{I}_\mathcal{D}) \neq T_\Pi^i(\mathfrak{I}_\mathcal{D})$, for each $i \in \{0, \ldots, k\}$. As each $T_\Pi^i(\mathfrak{I}_\mathcal{D})$ is a $(\Pi, \mathcal{D})$-interpretation (Wałęga et al., 2019), we obtain that each $T_\Pi^{i+1}(\mathfrak{I}_\mathcal{D})$ satisfies at least one more relational fact over a $(\Pi, \mathcal{D})$-interval than $T_\Pi^i(\mathfrak{I}_\mathcal{D})$. Thus, $T_\Pi^{k+1}(\mathfrak{I}_\mathcal{D})$ entails at least $k + 1$ such facts, and so does Π and $\mathcal{D}$, which raises a contradiction. □

4.2 Identifying the Relevant Segment of the Timeline

In this section, we show that all facts entailed by a finitely materialisable bounded program and a specific dataset hold within a single bounded interval.

We start by showing that facts entailed by an arbitrary (i.e., not necessarily finitely materialisable) bounded program Π and dataset $\mathcal{D}$ can be determined by considering (instead of $\mathcal{D}$) any dataset $\mathcal{D}_\varrho$ representing the projection of $\mathfrak{C}_{\Pi,\mathcal{D}}$ over a 'short' interval ϱ satisfying certain properties. This property is established by the following technical lemma.

Lemma 17. *Let Π and $\mathcal{D}$ be bounded, and let $\mathcal{D}_\varrho$ be a dataset representing the projection of $\mathfrak{C}_{\Pi,\mathcal{D}}$ over a closed interval ϱ of length $\mathsf{depth}(\Pi)$. The following hold:*

1. *If $\varrho^+ \geq t_\mathcal{D}^{\max}$ then, for each relational fact $M@t$ such that $t \geq \varrho^-$, it holds that $(\Pi, \mathcal{D}) \models M@t$ if and only if $(\Pi, \mathcal{D}_\varrho) \models M@t$,*
2. *If $\varrho^- \leq t_\mathcal{D}^{\min}$ then, for each relational fact $M@t$ such that $t \leq \varrho^+$, it holds that $(\Pi, \mathcal{D}) \models M@t$ if and only if $(\Pi, \mathcal{D}_\varrho) \models M@t$.*

Proof. To show Statement 1, assume that $\varrho^+ \geq t_\mathcal{D}^{\max}$ and that $M@t$ is a relational fact with $t \geq \varrho^-$. If $t \in \varrho$, then the statement follows from the fact that $\mathcal{D}_\varrho$ is a dataset representation of the projection of $\mathfrak{C}_{\Pi,\mathcal{D}}$ over ϱ. Next, assume that $t > \varrho^+$. As $(\Pi, \mathcal{D}) \models \mathcal{D}_\varrho$, we immediately obtain that $(\Pi, \mathcal{D}_\varrho) \models M@t$ implies $(\Pi, \mathcal{D}) \models M@t$. For the other implication it suffices to show that $T_\Pi^\alpha(\mathfrak{I}_\mathcal{D}) \models M'@t'$ implies $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}_\varrho}) \models M'@t'$, for every ordinal α and every relational fact $M'@t'$ with $t' > \varrho^+$. We proceed by a transfinite induction on ordinals α.

For the base case, we observe that $T_\Pi^0(\mathfrak{I}_\mathcal{D}) = \mathfrak{I}_\mathcal{D}$ and $\varrho^+ \geq t_\mathcal{D}^{\max}$. Since $\mathcal{D}$ is bounded and $t_\mathcal{D}^{\max}$ is the largest number mentioned in $\mathcal{D}$, no fact of the form $M'@t'$ with $t' > \varrho^+$ (and so, with $t' > t_\mathcal{D}^{\max}$) can be satisfied in $T_\Pi^0(\mathfrak{I}_\mathcal{D})$; therefore the implication holds trivially.

For the inductive step with a successor ordinal α, we assume that $T_\Pi^\alpha(\mathfrak{I}_\mathcal{D}) \models M'@t'$, where $M'@t'$ is a relational fact with $t' > \varrho^+$. Let $\mathfrak{I} = T_\Pi^{\alpha-1}(\mathfrak{I}_\mathcal{D})$, so $T_\Pi(\mathfrak{I}) \models M'@t'$. Hence, by Lemma 15, $T_\Pi(\mathfrak{I}|_{\varrho'}) \models M'@t'$, where $\varrho' = [t' - \mathsf{depth}(\Pi), t' + \mathsf{depth}(\Pi)]$. Since we need to show that $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}_\varrho}) \models M'@t'$, it suffices to prove that $\mathfrak{I}|_{\varrho'} \subseteq T_\Pi^{\alpha-1}(\mathfrak{I}_{\mathcal{D}_\varrho})$. In particular, we will show that $\mathfrak{I}|_{\varrho'} \subseteq \mathfrak{I}'|_{\varrho'}$, where $\mathfrak{I}' = T_\Pi^{\alpha-1}(\mathfrak{I}_{\mathcal{D}_\varrho})$. Observe that $\mathcal{D}_\varrho$ represents $\mathfrak{C}_{\Pi,\mathcal{D}}|_\varrho$,

so $\mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho} = \mathfrak{I}_{\mathcal{D}_\varrho}|_{\varrho}$. Moreover, $\mathfrak{I} \subseteq \mathfrak{C}_{\Pi,\mathcal{D}}$ and $\mathfrak{I}_{\mathcal{D}_\varrho} \subseteq \mathfrak{I}'$, so $\mathfrak{I}|_{\varrho} \subseteq \mathfrak{I}'|_{\varrho}$. By the inductive assumption, $\mathfrak{I}|_{(\varrho^+,\infty)} \subseteq \mathfrak{I}'|_{(\varrho^+,\infty)}$, so $\mathfrak{I}|_{[\varrho^-,\infty)} \subseteq \mathfrak{I}'|_{[\varrho^-,\infty)}$. Recall now that $t' > \varrho^+$ and $\varrho^+ - \varrho^- = \mathsf{depth}(\Pi)$, so $t' - \mathsf{depth}(\Pi) > \varrho^-$. As $t' - \mathsf{depth}(\Pi) = (\varrho')^-$, we obtain $(\varrho')^- > \varrho^-$, and so $\varrho' \subseteq [\varrho^-, \infty)$. Thus, $\mathfrak{I}|_{[\varrho^-,\infty)} \subseteq \mathfrak{I}'|_{[\varrho^-,\infty)}$ implies $\mathfrak{I}|_{\varrho'} \subseteq \mathfrak{I}'|_{\varrho'}$, as required.

Now, consider the inductive step for a limit ordinal α. If $T^\alpha_\Pi(\mathfrak{I}_\mathcal{D}) \models M'@t'$ then, by the fact that $T^\alpha_\Pi(\mathfrak{I}_\mathcal{D}) = \bigcup_{\beta<\alpha} T^\beta_\Pi(\mathfrak{I}_\mathcal{D})$, we obtain that $T^\beta_\Pi(\mathfrak{I}_\mathcal{D}) \models M'@t'$ for some $\beta < \alpha$. Thus, by the inductive assumption, we obtain that $T^\beta_\Pi(\mathfrak{I}_{\mathcal{D}_\varrho}) \models M'@t'$, and so $T^\alpha_\Pi(\mathfrak{I}_{\mathcal{D}_\varrho}) \models M'@t'$.

The proof for Statement 2 is analogous; in particular, we proceed by a transfinite induction on ordinals α to show that $T^\alpha_\Pi(\mathfrak{I}_\mathcal{D}) \models M'@t'$ implies $T^\alpha_\Pi(\mathfrak{I}_{\mathcal{D}_\varrho}) \models M'@t'$, for every relational fact $M'@t'$ with $t' < \varrho^-$. □

We next exploit the result in Lemma 17 to establish a sufficient condition for non-finite materialisability. In particular, given a bounded program Π and a dataset $\mathcal{D}$, if we can find two different intervals of a given length for which Π and $\mathcal{D}$ entail exactly the same atoms, then we can ensure that the contents of these intervals will also be entailed repeatedly throughout the infinite timeline. As a result, Π cannot be finitely materialisable for $\mathcal{D}$.

Lemma 18. *Let Π be a bounded program, let $\mathcal{D}$ be a bounded dataset, let ϱ_1, ϱ_2 be closed intervals of length $\mathsf{depth}(\Pi)$ such that $\varrho_1^+ < \varrho_2^+$, and let $q = \varrho_2^+ - \varrho_1^+$. Assume also that $\mathfrak{C}_{\Pi,\mathcal{D}} \models M@t$ if and only if $\mathfrak{C}_{\Pi,\mathcal{D}} \models M@t{+}q$, for each relational fact $M@t$ with $t \in \varrho_1$. Then, the following statements hold:*

1. *If $\varrho_1^+ \geq t^{\max}_\mathcal{D}$, then, for each relational fact $M@t$ with $t \geq \varrho_1^-$, we have $(\Pi,\mathcal{D}) \models M@t$ if and only if $(\Pi,\mathcal{D}) \models M@t{+}q$.*
2. *If $\varrho_2^- \leq t^{\min}_\mathcal{D}$, then, for each relational fact $M@t$ with $t \leq \varrho_2^+$, we have $(\Pi,\mathcal{D}) \models M@t$ if and only if $(\Pi,\mathcal{D}) \models M@t{-}q$.*

Proof. We focus on Statement 1 since the proof of the second statement is analogous. Hence, assume that $\varrho_1^+ \geq t^{\max}_\mathcal{D}$, and thus, $\varrho_2^+ > t^{\max}_\mathcal{D}$. Let $\mathcal{D}_{\varrho_1}$ and $\mathcal{D}_{\varrho_2}$ be datasets representing $\mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho_1}$ and $\mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho_2}$. Hence, $\mathcal{D}_{\varrho_1} \models M@t$ if and only if $\mathcal{D}_{\varrho_2} \models M@t{+}q$, for each relational fact $M@t$. By Lemma 17, $\mathfrak{C}_{\Pi,\mathcal{D}}|_{[\varrho_1^-,\infty)} = \mathfrak{C}_{\Pi,\mathcal{D}_{\varrho_1}}|_{[\varrho_1^-,\infty)}$ and $\mathfrak{C}_{\Pi,\mathcal{D}}|_{[\varrho_2^-,\infty)} = \mathfrak{C}_{\Pi,\mathcal{D}_{\varrho_2}}|_{[\varrho_2^-,\infty)}$. Therefore, the following six statements are equivalent for any metric atom $M@t$ with $t \geq \varrho_1^-$:

$$(\Pi,\mathcal{D}) \models M@t, \qquad \mathfrak{C}_{\Pi,\mathcal{D}}|_{[\varrho_1^-,\infty)} \models M@t, \qquad \mathfrak{C}_{\Pi,\mathcal{D}_{\varrho_1}}|_{[\varrho_1^-,\infty)} \models M@t,$$
$$\mathfrak{C}_{\Pi,\mathcal{D}_{\varrho_2}}|_{[\varrho_2^-,\infty)} \models M@t{+}q, \qquad \mathfrak{C}_{\Pi,\mathcal{D}}|_{[\varrho_2^-,\infty)} \models M@t{+}q, \qquad (\Pi,\mathcal{D}) \models M@t+q.$$

The first equivalence follows from the assumption that $t \geq \varrho_1^-$. The second and the fourth equivalence are a consequence of applying Lemma 17, whereas the third one results from $\mathcal{D}_{\varrho_2}$ being a shift of $\mathcal{D}_{\varrho_1}$. The last equivalence holds since $t + q \geq \varrho_2^-$. The equivalence of the first and the last statement proves Statement 1. □

We are now ready to identify the segment of the timeline over which facts can be entailed if the program and dataset of interest are finitely materialisable.

Theorem 19. *Let Π be a bounded program which is finitely materialisable for a bounded dataset $\mathcal{D}$, and let $\mathsf{offset}(\Pi,\mathcal{D}) = 2^{A \cdot B} \cdot \mathsf{depth}(\Pi)$ for A the number of relational atoms in $\mathsf{ground}(\Pi,\mathcal{D})$ and B the number of $(\Pi,\mathcal{D})$-intervals within $[t^{\max}_\mathcal{D}, t^{\max}_\mathcal{D} + \mathsf{depth}(\Pi)]$. Then, $t \in [t^{\min}_\mathcal{D} - \mathsf{offset}(\Pi,\mathcal{D}), t^{\max}_\mathcal{D} + \mathsf{offset}(\Pi,\mathcal{D})]$ for any relational fact $M@t$ entailed by Π and $\mathcal{D}$.*

Proof. Towards a contradiction, let us assume that Π and $\mathcal{D}$ entail a fact $M@t$ such that $t \notin [t^{\min}_{\mathcal{D}} - \mathsf{offset}(\Pi, \mathcal{D}), t^{\max}_{\mathcal{D}} + \mathsf{offset}(\Pi, \mathcal{D})]$. We assume that $t > t^{\max}_{\mathcal{D}} + \mathsf{offset}(\Pi, \mathcal{D})$, as the proof for $t < t^{\min}_{\mathcal{D}} - \mathsf{offset}(\Pi, \mathcal{D})$ is analogous. We obtain a contradiction by showing that Π and $\mathcal{D}$ entail infinitely many relational facts over $(\Pi, \mathcal{D})$-intervals. To this end, by Lemma 18, it suffices to show that there exist closed intervals ϱ_i, ϱ_j of length $\mathsf{depth}(\Pi)$ such that $t^{\max}_{\mathcal{D}} \leq \varrho_i^+ < \varrho_j^+$, $\mathfrak{C}_{\Pi,\mathcal{D}} \models M'@t'$ if and only if $\mathfrak{C}_{\Pi,\mathcal{D}} \models M'@t'+(\varrho_j^+ - \varrho_i^+)$, for each relational fact $M'@t'$ with $t' \in \varrho_i$, and such that $\mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho_i}$ is non-empty.

We claim that the required ϱ_i and ϱ_j must occur in the sequence $\varrho_1, \ldots, \varrho_{2^{A \cdot B}}$ of closed intervals of length $\mathsf{depth}(\Pi)$ such that $\varrho_1^- = t^{\max}_{\mathcal{D}}$ and $\varrho_k^+ = \varrho_{k+1}^-$, for all $k \in \{1, \ldots, 2^{A \cdot B} - 1\}$. We start by observing that $\mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho_k}$ is non-empty, for each ϱ_k. Otherwise, by Lemma 17, $\mathfrak{C}_{\Pi,\mathcal{D}}|_{[\varrho_k^-, \infty)}$ is empty, which raises a contradiction as $\varrho_{2^{A \cdot B}}^+ = t^{\max}_{\mathcal{D}} + \mathsf{offset}(\Pi, \mathcal{D})$, so $t > \varrho_{2^{A \cdot B}}^+$, and hence, $\mathfrak{C}_{\Pi,\mathcal{D}} \not\models M@t$. To finish the proof, we show that there exist ϱ_i, ϱ_j in our sequence of intervals such that $\mathfrak{C}_{\Pi,\mathcal{D}} \models M'@t'$ if and only if $\mathfrak{C}_{\Pi,\mathcal{D}} \models M'@t'+(\varrho_j^+ - \varrho_i^+)$, for each relational fact $M'@t'$ with $t' \in \varrho_i$. First, observe that, by the definitions of the $(\Pi, \mathcal{D})$-ruler and $\mathsf{depth}(\Pi)$, the nth $(\Pi, \mathcal{D})$-interval in ϱ_i has the same length as the nth $(\Pi, \mathcal{D})$-interval in ϱ_j, for each $n \leq B$ and $i, j \leq 2^{A \cdot B}$. Moreover, as each ϱ_k in our sequence contains B $(\Pi, \mathcal{D})$-intervals and, in each of these $(\Pi, \mathcal{D})$-intervals, Π and $\mathcal{D}$ entail one of 2^A sets of relational atoms, over each ϱ_k in the sequence Π and $\mathcal{D}$ entail one of $2^{A \cdot B}$ combinations of sets of relational atoms. However, as we have shown, within each ϱ_k some fact must hold, so the longest sequence of intervals in which required ϱ_i, ϱ_j do not occur has $2^{A \cdot B} - 1$ elements. Our sequence has $2^{A \cdot B}$ elements, so the required pair of intervals must exist. □

4.3 A Materialisation-based Algorithm for Checking Finite Materialisability

The result in Theorem 19 suggests the decision procedure for checking finite materialisability specified by Algorithm 3; when given a bounded program Π and a bounded dataset $\mathcal{D}$ as input, the algorithm performs materialisation in the usual way (c.f. Algorithm 2) until a fixpoint is reached, in which case a full materialisation has been computed and thus Π is finitely materialisable for $\mathcal{D}$, or until some fact is derived outside the interval $\varrho = t \in [t^{\min}_{\mathcal{D}} - \mathsf{offset}(\Pi, \mathcal{D}), t^{\max}_{\mathcal{D}} + \mathsf{offset}(\Pi, \mathcal{D})]$. In the latter case, Theorem 19 ensures that Π is not finitely materialisable for $\mathcal{D}$.

The running time of Algorithm 3 is determined by the number of facts that can be derived via rule application, which also bounds the number of iterations of the main loop. We show that the algorithm can derive at most a doubly exponential number of facts before stopping; in particular, the number of derived facts is proportional to the size of the grounding of Π with respect to $\mathcal{D}$ (of exponential size) and the number of $(\Pi, \mathcal{D})$-intervals in ϱ (which is of doubly exponential size given that the length of ϱ is also doubly exponential).

Theorem 20. *Algorithm 3 returns* true *if the input program* Π *is finitely materialisable for the input dataset* $\mathcal{D}$*, and it returns* false *otherwise. Moreover, Algorithm 3 works in doubly exponential time in the size of* Π *and* $\mathcal{D}$*.*

Proof. Correctness of the algorithm follows from Theorem 19. To show that it terminates in doubly exponential time in the size of Π and $\mathcal{D}$ given as input, we will first show that the number of iterations of the loop in Lines 3–7 is at most doubly exponential and, secondly, that each iteration is feasible in doubly exponential time.

Algorithm 3: Checking finite materialisability for a single dataset

Input: A bounded program Π and a bounded dataset $\mathcal{D}$
Output: A boolean value true if Π is finitely materialisable for $\mathcal{D}$, false otherwise

1 $\mathcal{D}_{\mathsf{new}} \coloneqq \mathcal{D}$;
2 $\varrho \coloneqq [t^{\min}_{\mathcal{D}} - \mathsf{offset}(\Pi, \mathcal{D}), t^{\max}_{\mathcal{D}} + \mathsf{offset}(\Pi, \mathcal{D})]$;
3 **repeat**
4 $\quad \mathcal{D}_{\mathsf{old}} \coloneqq \mathcal{D}_{\mathsf{new}}$;
5 $\quad \mathcal{D}_{\mathsf{new}} \coloneqq \mathsf{ApplyRules}(\Pi, \mathcal{D}_{\mathsf{old}})$;
6 $\quad$ **if** *there is* $M@\varrho' \in \mathcal{D}_{\mathsf{new}}$ *with* $\varrho' \not\subseteq \varrho$ **then return** false;
7 **until** $\mathfrak{I}_{\mathcal{D}_{\mathsf{old}}} = \mathfrak{I}_{\mathcal{D}_{\mathsf{new}}}$;
8 **return** true;

The stopping conditions in Lines 6 and 7 ensure that the number of iterations of the loop from Lines 3–7 is bounded by the maximal number of facts that are stored in $\mathcal{D}_{\mathsf{old}}$ in the run of the algorithm. Since $\mathcal{D}_{\mathsf{old}}$ represents $T^i_\Pi(\mathfrak{I}_\mathcal{D})$, for some $i \in \mathbb{N}$, we obtain that each $M@\varrho' \in \mathcal{D}_{\mathsf{old}}$ is such that M occurs in $\mathsf{ground}(\Pi, \mathcal{D})$ and the endpoints of ϱ' belong to the $(\Pi, \mathcal{D})$-ruler, as shown by Wałęga et al. (2019, Lemma 4). Moreover, the stopping condition from Line 6 ensures that $\varrho' \subseteq \varrho$, and since $\mathcal{D}_{\mathsf{old}}$ is constructed by calling (i times) $\mathsf{ApplyRules}$, all facts in $\mathcal{D}_{\mathsf{old}}$ are coalesced. Thus the number of facts in $\mathcal{D}_{\mathsf{old}}$ is bounded by $A \cdot C$, where A is the number of relational atoms in $\mathsf{ground}(\Pi, \mathcal{D})$ and C is the number of $(\Pi, \mathcal{D})$-intervals contained in ϱ. Clearly, A is exponentially large (in the size of the representations of Π and $\mathcal{D}$). Next, we show that C is doubly exponential. To this end, observe that $\varrho = [t^{\min}_{\mathcal{D}} - \mathsf{offset}(\Pi, \mathcal{D}), t^{\max}_{\mathcal{D}} + \mathsf{offset}(\Pi, \mathcal{D})]$, and $\mathsf{offset}(\Pi, \mathcal{D}) = 2^{A \cdot B} \cdot \mathsf{depth}(\Pi)$, for B the number of $(\Pi, \mathcal{D})$-intervals in $[t^{\max}_{\mathcal{D}}, t^{\max}_{\mathcal{D}} + \mathsf{depth}(\Pi)]$. Thus, $C \leq 2 \cdot (2^{A \cdot B} \cdot (B - 1)) + E$, where E is the number of $(\Pi, \mathcal{D})$-intervals contained in $[t^{\min}_{\mathcal{D}}, t^{\max}_{\mathcal{D}}]$. We can observe that $B \leq 2 \cdot |\mathcal{D}| \cdot \frac{\mathsf{depth}(\Pi)}{\mathsf{div}(\Pi)} + 1$, as $[t, t + \mathsf{div}(\Pi))$ (for any t on the $(\Pi, \mathcal{D})$-ruler) contains at most $2 \cdot |\mathcal{D}|$ $(\Pi, \mathcal{D})$-intervals. Due to the binary encoding of numbers in Π, both the value of $\mathsf{depth}(\Pi)$ and the value of $\frac{1}{\mathsf{div}(\Pi)}$ are at most exponentially large, and so is B. Moreover, E is also exponential, as $E \leq 2 \cdot |\mathcal{D}| \cdot \left\lceil \frac{t^{\max}_{\mathcal{D}} - t^{\min}_{\mathcal{D}}}{\mathsf{div}(\Pi)} \right\rceil + 1$. Consequently, C is at most doubly exponentially large, and so, the number of iterations of the loop from Lines 3–7 is at most doubly exponential.

Next, we show that each iteration of the loop is feasible in doubly exponential time. To this end, we observe that $\mathcal{D}_{\mathsf{old}}$ not only has doubly exponentially many facts but also that all intervals mentioned in $\mathcal{D}_{\mathsf{old}}$ are doubly exponentially representable (this is so because they have endpoints on the $(\Pi, \mathcal{D})$-ruler and are contained in ϱ). Hence, the construction in Line 5 of a dataset representing $T_\Pi(\mathfrak{I}_{\mathcal{D}_{\mathsf{old}}})$, is performed by $\mathsf{ApplyRules}$ from Algorithm 1 in doubly exponential time. Indeed, the loop in Lines 2–9 of Algorithm 1 is traversed at most as many times as there are rules in $\mathsf{ground}(\Pi, \mathcal{D}_{\mathsf{old}})$, which coincides with $\mathsf{ground}(\Pi, \mathcal{D})$. The number of these rules is exponentially bounded; moreover, since $\mathcal{D}_{\mathsf{old}}$ is doubly exponentially large, each iteration of the loop in Lines 2–9 is computed in doubly exponential time. Finally, we observe that the stopping conditions in Lines 6 and 7 of Algorithm 3 can also be checked within doubly exponential time. □

4.4 Complexity of Checking Data-dependent Finite Materialisability

Algorithm 3 can be easily incorporated into existing DatalogMTL reasoners with a materialisation component, such as MeTeoR (Wang et al., 2022). Algorithm 3 is, however, not worst-case optimal, as we show in this section. In particular, we show that the problem of checking whether a given bounded program is finitely materialisable for a given bounded dataset is ExpSpace-complete for combined complexity and PSpace-complete for data complexity; thus, it is no harder than fact entailment for the bounded fragment of DatalogMTL (c.f. Theorems 46 and 47). Our upper bounds do not immediately lead to a practical algorithm since they rely on reductions to fact entailment in DatalogMTL extended with stratified negation-as-failure, for which all known decision procedures are automata-based (Tena Cucala et al., 2021).

Theorem 21. *Checking if a bounded program is finitely materialisable for a given bounded dataset is* PSpace*-complete for data complexity.*

Proof. For the upper bound, we will reduce our problem to fact entailment in stratifiable DatalogMTL$^\neg$ programs, which is known to be PSpace-complete in data complexity (Tena Cucala et al., 2021, Theorem 19). Stratifiable programs allow for negation-as-failure in rule bodies, denoted with not, while satisfying standard stratification conditions. We construct a stratifiable program Π' by extending Π in a data-independent way with the following rules, where R, L, and G are fresh predicates:

$$R \leftarrow \diamondminus_{[0,\infty)} P(\mathbf{x}), \qquad \text{for every predicate } P \text{ occurring in } \Pi$$
$$L \leftarrow \diamondplus_{[0,\infty)} P(\mathbf{x}), \qquad \text{for every predicate } P \text{ occurring in } \Pi$$
$$G \leftarrow \mathsf{not}\ \boxminus_{[0,\infty)} R \wedge \mathsf{not}\ \boxplus_{[0,\infty)} L.$$

We show that, for each dataset $\mathcal{D}$, program Π is finitely materialisable for $\mathcal{D}$ if and only if Π' and $\mathcal{D}$ entail $G@0$. To this end, let ϱ be the minimal interval containing all the time points over which Π and $\mathcal{D}$ entail some relational fact with a predicate occurring in Π. By the first two types of the new rules in Π', we obtain that R holds at all $t > \varrho^-$ and L holds at all $t < \varrho^+$, respectively. Hence, by the last rule, G holds at each time point on the timeline, including 0, if there exists a time point in which R does not hold and a time point in which L does not hold, that is, if ϱ is bounded from the left and from the right; otherwise G does not hold at any time point. Of course, if ϱ is bounded, there are finitely many $(\Pi, \mathcal{D})$-intervals contained in ϱ, and so, only finitely many relational facts can be entailed by Π and $\mathcal{D}$ over such intervals. By the assumption, Π and $\mathcal{D}$ do not entail any facts outside ϱ, therefore, the boundedness of ϱ implies that Π and $\mathcal{D}$ entail finitely many relational facts over $(\Pi, \mathcal{D})$-intervals. Consequently, the following statements are equivalent:

1. Π' and $\mathcal{D}$ entail $G@0$,
2. there exist time points $t \leq t'$ such that $(\Pi', \mathcal{D}) \not\models R@t$ and $(\Pi', \mathcal{D}) \not\models L@t'$,
3. there exist time points $t \leq t'$ such that $\varrho \subseteq [t, t']$,
4. Π and $\mathcal{D}$ entail finitely many relational facts over $(\Pi, \mathcal{D})$-intervals.

To complete the proof it suffices to observe that, by Theorem 16, Statement 4 is equivalent to the fact that Π is finitely materialisable for $\mathcal{D}$.

For the lower bound, we provide a reduction from the acceptance problem for a deterministic Turing machine running in polynomial space. Let $\mathfrak{M} = (\Sigma, \mathcal{Q}, \delta, q_{init}, q_{acc}, q_{rej})$ be such a machine with alphabet Σ, states $\mathcal{Q}$ including an initial state q_{init}, accepting state q_{acc}, and rejecting state q_{rej}, and transitions $\delta : \Sigma_\sqcup \times (\mathcal{Q} \setminus \{q_{acc}, q_{rej}\}) \mapsto \Sigma_\sqcup \times \mathcal{Q} \times \{\mathsf{L}, \mathsf{R}\}$ where $\Sigma_\sqcup$ is alphabet Σ augmented with the blank symbol $\sqcup$. We let p be a polynomial such that, for every input word w, machine $\mathfrak{M}$ uses at most $p(|w|)$ cells for computations.

We now define a bounded program $\Pi_\mathfrak{M}$ and construct in logarithmic space a dataset $\mathcal{D}_w$ from a word w, so that $\mathfrak{M}$ accepts w if and only if $\Pi_\mathfrak{M}$ is not finitely materialisable for $\mathcal{D}_w$. The main idea is to encode the ith configuration of $\mathfrak{M}$ on w using facts holding at the time point i. These facts will use constants $c_1, \dots, c_{p(|w|)}$ representing tape cells, three types of unary predicates, namely, *Head*, *NoHead*, and $Cont_s$, for $s \in \Sigma_\sqcup$, as well as propositions $State_q$, for $q \in \mathcal{Q}$. These predicates describe the position of the head, the contents of the tape, and the state of the machine in consecutive configurations. We will also use binary predicates *Next* and *Neq* to state that pairs of cells are adjacent or distinct, respectively. Furthermore, we employ the nullary predicate *Tape* to delimit the segment of the timeline that simulates the machine's computations. Finally, we use a proposition G to indicate that the machine has entered the accepting state.

We define $\Pi_\mathfrak{M}$ as a bounded program consisting of the following rules, where $X \in \{\mathsf{L}, \mathsf{R}\}$ indicates direction of $\mathfrak{M}$'s head movement:

$$
\begin{aligned}
\boxplus_1 Cont_{s'}(x) &\leftarrow Cont_s(x) \wedge State_q \wedge Head(x) \wedge Tape, && \text{for each } \delta(s,q) = (s', q', X),\\
\boxplus_1 State_{q'} &\leftarrow Cont_s(x) \wedge State_q \wedge Head(x) \wedge Tape, && \text{for each } \delta(s,q) = (s', q', X),\\
\boxplus_1 Head(y) &\leftarrow Cont_s(x) \wedge State_q \wedge Head(x) \wedge Next(y,x) \wedge Tape, && \text{for each } \delta(s,q) = (s', q', \mathsf{L}),\\
\boxplus_1 Head(y) &\leftarrow Cont_s(x) \wedge State_q \wedge Head(x) \wedge Next(x,y) \wedge Tape, && \text{for each } \delta(s,q) = (s', q', \mathsf{R}),\\
\boxplus_1 Cont_s(x) &\leftarrow Cont_s(x) \wedge NoHead(x) \wedge Tape, && \text{for each } s \in \Sigma_\sqcup,\\
NoHead(y) &\leftarrow Head(x) \wedge Neq(x,y) \wedge Tape,\\
G &\leftarrow State_{q_{acc}} \wedge Tape,\\
\boxminus_1 G &\leftarrow G.
\end{aligned}
$$

The first four types of rules simulate transitions of the machine. Rules of the fifth type capture inertia and copy over the contents of cells not affected by a transition (i.e., such that the head is not located over them), whereas the sixth rule states that $NoHead(c)$ holds for those cells c for which $Head(c)$ does not hold. The last two rules allow us to derive G when the accepting state is reached, and then, propagate G to all preceding integer time points (in particular, to the time point 1).

Next, we show how to construct a dataset $\mathcal{D}_w$ based on an input word $w = s_1 \dots s_n$. To this end, we let $m = |\Sigma_\sqcup|^{p(n)} \cdot |\mathcal{Q}| \cdot p(n)$ be the number of possible configurations of $\mathfrak{M}$.

Then, $\mathcal{D}_w$ consists of the following facts:

$$
\begin{aligned}
&\mathit{Cont}_{s_i}(c_i)@1, && \text{for each } i \in \{1, \dots, n\},\\
&\mathit{Cont}_{\sqcup}(c_i)@1, && \text{for each } i \in \{n+1, \dots, p(n)\},\\
&\mathit{State}_{q_{\mathit{init}}}@1,\\
&\mathit{Head}(c_1)@1,\\
&\mathit{Next}(c_i, c_{i+1})@[1, m], && \text{for each } i \in \{1, \dots, p(n)-1\},\\
&\mathit{Neq}(c_i, c_j)@[1, m], && \text{for all } i, j \in \{1, \dots, p(n)\} \text{ such that } i \neq j,\\
&\mathit{Tape}@[1, m].
\end{aligned}
$$

We observe that $\mathcal{D}_w$ can be computed in logarithmic space; in particular, the number $m = |\Sigma_{\sqcup}|^{p(n)} \cdot |\mathcal{Q}| \cdot p(n)$, occurring as an endpoint of some intervals in $\mathcal{D}_w$, can be computed in TC^0 because powering binary numbers is feasible in TC^0 (Reif & Tate, 1992; Immerman & Landau, 1989).

First we show that the canonical interpretation $\mathfrak{C}_{\Pi_{\mathfrak{M}}, \mathcal{D}_w}$ encodes the computation of $\mathfrak{M}$ on w. In particular, by induction on computation steps $t \in \{1, \dots, m\}$ in the run, we can simultaneously prove that the following statements hold for every cell number $i \in \{1, \dots, p(n)\}$, for every symbol $s \in \Sigma_{\sqcup}$, and for every state $q \in \mathcal{Q}$:

- $\mathfrak{C}_{\Pi_{\mathfrak{M}}, \mathcal{D}_w} \models \mathit{Cont}_s(c_i)@t$ if and only if s is the content of the ith cell at the step t,
- $\mathfrak{C}_{\Pi_{\mathfrak{M}}, \mathcal{D}_w} \models \mathit{State}_q@t$ if and only if q is the state at the step t,
- $\mathfrak{C}_{\Pi_{\mathfrak{M}}, \mathcal{D}_w} \models \mathit{Head}(c_i)@t$ if and only if the head is over the ith cell at the step t,
- $\mathfrak{C}_{\Pi_{\mathfrak{M}}, \mathcal{D}_w} \models \mathit{NoHead}(c_i)@t$ if and only if the head is not over the ith cell at the step t.

Next, we use the above properties to show that $\mathfrak{M}$ accepts a word w if and only if $\Pi_{\mathfrak{M}}$ is not finitely materialisable for $\mathcal{D}_w$. For the forward direction observe that the following statements are equivalent:

1. $\mathfrak{M}$ enters the accepting state in the run on w,
2. $\mathfrak{C}_{\Pi_{\mathfrak{M}}, \mathcal{D}_w} \models \mathit{State}_{q_{\mathit{acc}}}@t$, for some $t \in \{1, \dots, m\}$,
3. $\mathfrak{C}_{\Pi_{\mathfrak{M}}, \mathcal{D}_w} \models G@t'$, for all $t' \leq t$ and some $t \in \{1, \dots, m\}$.

Note also that Statement 3 implies, by Theorem 16, that $\Pi_{\mathfrak{M}}$ is not finitely materialisable for $\mathcal{D}_{\mathfrak{M}}$, as required.

For the opposite direction, assume that $\mathfrak{M}$ does not accept w. To prove that $\Pi_{\mathfrak{M}}$ is finitely materialisable for $\mathcal{D}_w$, it suffices to show that $\Pi_{\mathfrak{M}}$ and $\mathcal{D}_w$ entail finitely many facts over $(\Pi_{\mathfrak{M}}, \mathcal{D}_w)$-intervals. To this end, we will show that if $\Pi_{\mathfrak{M}}$ and $\mathcal{D}_w$ entail a relational fact $M@t$, then $t \in [1, m+1]$. First, observe that M cannot be G; indeed, since $\mathfrak{M}$ does not accept w, for all t', we have $\mathfrak{C}_{\Pi_{\mathfrak{M}}, \mathcal{D}_w} \not\models \mathit{State}_{q_{\mathit{acc}}}@t'$ and $\mathfrak{C}_{\Pi_{\mathfrak{M}}, \mathcal{D}_w} \not\models G@t'$. Next, observe that all facts in $\mathcal{D}_w$ are over intervals contained in $[1, m]$ and all rules of $\Pi_{\mathfrak{M}}$ which do not mention G in the head have a body atom Tape. Since Tape holds only in $[1, m]$ and the only temporal operator occurring in heads of those rules in $\Pi_{\mathfrak{M}}$ which do not mention G is $\boxplus_1$, we get $t \in [1, m+1]$. Hence PSpace-hardness follows. □

Next we provide tight bounds for combined complexity.

Theorem 22. *Checking if a bounded program is finitely materialisable for a given bounded dataset is* ExpSpace*-complete for combined complexity.*

Proof. For the upper bound, we observe that fact entailment in stratifiable DatalogMTL$^\neg$ is ExpSpace-complete for combined complexity (Tena Cucala et al., 2021, Theorem 20). Hence, we can use the exact same reduction as in the proof of the upper bound in Theorem 21, which immediately yields the ExpSpace upper bound.

For the lower bound, we modify the reduction of halting of deterministic Turing machines with an exponentially long tape, provided by Brandt et al. (2018, Theorem 8). Given a Turing machine $\mathfrak{M}$ and an input word w, they constructed a program Π and a dataset $\mathcal{D}$ such that $\mathfrak{M}$ halts on w if and only if Π and $\mathcal{D}$ are inconsistent. In particular, they used rules $\bot \leftarrow H_{q_h,a}$ to obtain inconsistency whenever the machine enters the halting state q_h and the head of the machine is over a cell with some tape symbol a.

The dataset $\mathcal{D}$ constructed by Brandt et al. is bounded, but their program Π includes rules with $\bot$ in the head as well as a rule mentioning unbounded interval, both of which are disallowed in bounded programs. Thus we start by constructing a bounded program Π' which behaves similarly to Π. To this end, we delete from Π all rules $\bot \leftarrow H_{q_h,a}$. We observe that the only remaining non-bounded rule in Π is of the form $\boxplus_1 N_\# \leftarrow N_\# \wedge \diamondplus_{(0,\infty)} N_\#^<$. This rule, together with the facts $N_\#@[m+1, m+1]$ and $N_\#^<@[2^n, 2^n]$ in $\mathcal{D}$, is responsible for entailing $N_\#$ in all integer time points between $m+1$ and 2^n. We obtain the same behaviour by replacing this rule with a bounded rule $\boxplus_1 N_\# \leftarrow N_\# \wedge \diamondplus_{(0,2^n)} N_\#^<$, which finalises our construction of Π'. Similarly to the reduction of Brandt et al., we obtain that $\mathfrak{M}$ halts on w if and only if Π' and $\mathcal{D}$ entail $H_{q_h,a}@t$, for some a and t.

Our goal however is to provide a reduction to finite materialisability and not to fact entailment. Thus, we further modify Π' in two steps to obtain the final program Π''. First, for every rule r in Π' and every pair consisting of a non-halting state q and an alphabet symbol a, we add an atom $\diamondplus_{[0,2^n+1]} H_{q,a}$ to the body of r. It guarantees that if the halting state q_h is reached by the Turing machine—and so $H_{q_h,a}@t$ is entailed, for some a and t—then no rule of the program can be applied at time points greater than $t + 2^n + 1$. Furthermore, by the form of the program and the dataset, no facts can be entailed in time points smaller than 1. Thus, our so-far constructed program is finitely materialisable for $\mathcal{D}$. To construct the final program Π'', however, we also add rules $\boxplus_{[0,1]} \boxminus_{[0,1]} H_{q_h,a} \leftarrow H_{q_h,a}$, for any alphabet symbol a, which recursively propagate $H_{q_h,a}$ to all time points. Thus, we obtain that Π'' and $\mathcal{D}$ entail infinitely many facts over $(\Pi'', \mathcal{D})$-intervals (and so, Π'' is not finitely materialisable for $\mathcal{D}$) if and only if $(\Pi, \mathcal{D}) \models H_{q_h,a}@t$, for some a and t, which is equivalent to halting of $\mathfrak{M}$ on w.

Therefore, we have reduced non-halting of a deterministic Turing machine with an exponential tape to finite materialisability. Importantly, our transformation of Π to Π'' is polynomial; in particular, the number 2^n occurring in one of the new rules in Π'' is polynomially representable since numbers in DatalogMTL programs are encoded in binary. As ExpSpace is closed under the complement, we obtained the required ExpSpace lower bound. □

5. Data-independent Finite Materialisability

In this section, we study the data-independent notion of finite materialisability. In particular, given a bounded program as input, our aim is to check whether the program is finitely materialisable for *every* bounded dataset.

To this end, we start by showing in Section 5.1 that it suffices to check finite materialisability of the input program for a single *critical dataset*. The idea of constructing a critical dataset has been exploited to establish both decidability and undecidability results for a number of data-independent reasoning problems in Database Theory, such as that of deciding universal termination of variants of the chase procedure for various extensions of Datalog (Gogacz & Marcinkowski, 2014; Cuenca Grau et al., 2013; Marnette & Geerts, 2010). These, as well as other related works, will be discussed in more detail in Section 8.

As a result, data-independent finite materialisability reduces to its data-dependent counterpart studied in Section 4. In particular, the materialisation-based algorithm in Section 4.3 can still be applied to the data-independent setting by replacing the input dataset of the algorithm with the critical dataset, which is dependent only on the input program. This approach, however, yields a decision procedure for the data-independent problem that is still doubly exponential in the size of the input program. In Section 5.2, we show that the relevant segment of the timeline identified in Section 4.2 for programs that are finitely materialisable for a given dataset can be further refined if we additionally know that the input program is finitely materialisable for *all* bounded datasets.

Based on this result, we propose in Section 5.3 a refined materialisation-based algorithm that solves the data-independent version of finite materialisability in singly exponential time. This upper bound is tight and hence we conclude that the problem has the same complexity as fact entailment in standard (non-temporal) Datalog and our algorithm for solving the problem is worst-case optimal.

5.1 The Critical Dataset

In this section, we show that the data-independent version of the finite materialisability problem for bounded programs can be solved by focusing on a specific critical dataset. The non-temporal part of the critical dataset for a bounded program Π follows the standard construction used in Database Theory for showing universal termination of the chase in the context of extensions of Datalog with existential quantification in the head of rules; in particular, we consider all relational facts that can be constructed with constants occurring in Π plus a single fresh constant. For the temporal part, the critical dataset uses a single interval of length $\mathsf{depth}(\Pi)$ for all facts in the dataset.

Definition 23. *The* critical dataset $\mathcal{D}_\Pi$ *for a bounded program* Π *is the set of all relational facts* $P(\mathbf{s})@[0, \mathsf{depth}(\Pi)]$ *such that* P *is a predicate mentioned in* Π *and each term in* $\mathbf{s}$ *is either a constant mentioned in* Π *or a single fresh constant* c_Π.

The choice of the interval $[0, \mathsf{depth}(\Pi)]$ is justified by the characterisation of finite materialisability in Theorem 16 and the statements in Lemma 17, which show that if Π and a dataset $\mathcal{D}$ entail infinitely many relational facts over $(\Pi, \mathcal{D})$-intervals, then so do Π and any dataset $\mathcal{D}'$ representing a projection of the canonical interpretation over some interval

of length $\mathsf{depth}(\Pi)$; thus, it suffices to include in the critical dataset facts over any interval of length $\mathsf{depth}(\Pi)$, and we chose $[0, \mathsf{depth}(\Pi)]$ as such an interval.

Theorem 24. *A bounded program Π is finitely materialisable for the class of bounded datasets if and only if Π is finitely materialisable for $\mathcal{D}_\Pi$.*

Proof. The forward direction holds trivially as $\mathcal{D}_\Pi$ is a bounded dataset. We show the opposite direction by contraposition. Assume that there is a bounded dataset $\mathcal{D}$ for which Π is not finitely materialisable. By Theorem 16, Π and $\mathcal{D}$ entail infinitely many relational facts over $(\Pi, \mathcal{D})$-intervals. Hence, either Π and $\mathcal{D}$ entail infinitely many relational facts over $(\Pi, \mathcal{D})$-intervals located to the right of $t^{\max}_{\mathcal{D}}$, or infinitely many such facts to the left of $t^{\min}_{\mathcal{D}}$. We focus on the first case, as the second one is proved analogously. We transform $\mathcal{D}$ to obtain datasets defined as follows, where we let $M[c_\Pi]$ be an atom obtained from M by replacing all constants not occurring in Π with a single fresh constant c_Π:

$$\begin{aligned}
\mathcal{D}_1 &= \{M@\varrho \mid M@\varrho \in \mathcal{D} \text{ and the predicate occurring in } M \text{ is mentioned in } \Pi\},\\
\mathcal{D}_2 &= \{M@\varrho \mid \mathfrak{C}_{\Pi,\mathcal{D}} \models M@\varrho, M \text{ is a relational atom in } \mathsf{ground}(\Pi,\mathcal{D}), \text{ and}\\
&\qquad\qquad \varrho \subseteq [t^{\max}_{\mathcal{D}} - \mathsf{depth}(\Pi), t^{\max}_{\mathcal{D}}] \text{ is a } (\Pi,\mathcal{D})\text{-interval}\},\\
\mathcal{D}_3 &= \{M@[t^{\max}_{\mathcal{D}} - \mathsf{depth}(\Pi), t^{\max}_{\mathcal{D}}] \mid M \text{ is a relational atom in } \mathsf{ground}(\Pi,\mathcal{D})\},\\
\mathcal{D}_4 &= \{M@[0, \mathsf{depth}(\Pi)] \mid M \text{ is a relational atom in } \mathsf{ground}(\Pi,\mathcal{D})\},\\
\mathcal{D}_5 &= \{M[c_\Pi]@\varrho \mid M@\varrho \in \mathcal{D}_4\}.
\end{aligned}$$

We will show that Π and each $\mathcal{D}_i$ entail infinitely many relational facts over $(\Pi, \mathcal{D})$-intervals. This completes the proof since $\mathcal{D}_5 = \mathcal{D}_\Pi$, and so, Π and $\mathcal{D}_\Pi$ entail infinitely many relational facts over $(\Pi, \mathcal{D}_\Pi)$-intervals, implying that Π is not finitely materialisable for $\mathcal{D}_\Pi$.

We start by observing that since Π and $\mathcal{D}$ entail infinitely many relational facts over $(\Pi, \mathcal{D})$-intervals, then so do Π and $\mathcal{D}_1$, because facts in $\mathcal{D}$ with predicates not mentioned in Π cannot be used to derive any new facts. By our previous assumption, these infinitely many facts are entailed to the right of $t^{\max}_{\mathcal{D}}$, so the same holds for Π and $\mathcal{D}_1$. Hence, $\mathfrak{C}_{\Pi,\mathcal{D}_1}|_{[t^{\max}_{\mathcal{D}} - \mathsf{depth}(\Pi), \infty)}$ also satisfies infinitely many relational facts over $(\Pi, \mathcal{D})$-intervals. We observe that $\mathcal{D}_2$ represents the projection of $\mathfrak{C}_{\Pi,\mathcal{D}_1}$ over $[t^{\max}_{\mathcal{D}} - \mathsf{depth}(\Pi), t^{\max}_{\mathcal{D}}]$. Thus, $\mathfrak{C}_{\Pi,\mathcal{D}_1}|_{[t^{\max}_{\mathcal{D}} - \mathsf{depth}(\Pi), \infty)} = \mathfrak{C}_{\Pi,\mathcal{D}_2}|_{[t^{\max}_{\mathcal{D}} - \mathsf{depth}(\Pi), \infty)}$ by Statement 1 in Lemma 17. Therefore, Π and $\mathcal{D}_2$ also entail infinitely many relational facts over $(\Pi, \mathcal{D})$-intervals. Next, observe that $\mathcal{D}_3 \models \mathcal{D}_2$ and hence Π and $\mathcal{D}_3$ must also entail infinitely many relational facts over $(\Pi, \mathcal{D})$-intervals. The same holds also for $\mathcal{D}_4$, which is obtained by shifting all facts in $\mathcal{D}_3$ by $-t^{\max}_{\mathcal{D}} + \mathsf{depth}(\Pi)$. To prove our claim for $\mathcal{D}_5$, it suffices to show that, for every ordinal α and for every relational fact $M@t$, if $T^\alpha_\Pi(\mathfrak{I}_{\mathcal{D}_4}) \models M@t$, then $T^\alpha_\Pi(\mathfrak{I}_{\mathcal{D}_5}) \models M[c_\Pi]@t$. We show this by transfinite induction on ordinals α.

The base case holds trivially since $\mathfrak{I}_{\mathcal{D}_4} \models M@t$ implies $\mathfrak{I}_{\mathcal{D}_5} \models M[c_\Pi]@t$ directly by the definition of $\mathcal{D}_5$. Assume now that $T^\alpha_\Pi(\mathfrak{I}_{\mathcal{D}_4}) \models M@t$, for α a successor ordinal. If $T^{\alpha-1}_\Pi(\mathfrak{I}_{\mathcal{D}_4}) \models M@t$, then, by the inductive assumption, $T^{\alpha-1}_\Pi(\mathfrak{I}_{\mathcal{D}_5}) \models M[c_\Pi]@t$, and so, $T^\alpha_\Pi(\mathfrak{I}_{\mathcal{D}_5}) \models M[c_\Pi]@t$. Next assume that $T^{\alpha-1}_\Pi(\mathfrak{I}_{\mathcal{D}_4}) \not\models M@t$, so there must exist a rule $r \in \mathsf{ground}(\Pi,\mathcal{D})$, say of the form $M' \leftarrow M_1 \wedge \cdots \wedge M_n$, and a time point t' such that all the atoms $M_1, \ldots, M_n$ are satisfied in $T^{\alpha-1}_\Pi(\mathfrak{I}_{\mathcal{D}_4})$ at t', and moreover, $M'@t' \models M@t$. By the inductive assumption, $M_1[c_\Pi], \ldots, M_n[c_\Pi]$ need to be satisfied in $T^{\alpha-1}_\Pi(\mathfrak{I}_{\mathcal{D}_5})$ at t'.

Hence, a single application of T_Π to $T_\Pi^{\alpha-1}(\mathfrak{I}_{\mathcal{D}_5})$, allows us to derive $M'[c_\Pi]@t'$, that is, $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}_5}) \models M'[c_\Pi]@t'$. However, $M'@t' \models M@t$ implies that $M'[c_\Pi]@t' \models M[c_\Pi]@t$, and so, $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}_5}) \models M[c_\Pi]@t$. Finally, assume $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}_4}) \models M@t$, for α a limit ordinal. Since $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}_4}) = \bigcup_{\beta<\alpha} T_\Pi^\beta(\mathfrak{I}_{\mathcal{D}_4})$, there exists $\beta < \alpha$ such that $T_\Pi^\beta(\mathfrak{I}_{\mathcal{D}_4}) \models M@t$. Then, by the inductive assumption, we obtain that $T_\Pi^\beta(\mathfrak{I}_{\mathcal{D}_5}) \models M[c_\Pi]@t$, and so $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}_5}) \models M[c_\Pi]@t$. □

5.2 Revising the Relevant Segment of the Timeline

In Section 4.2, we showed that all facts entailed by a bounded program Π and dataset $\mathcal{D}$ must hold within a single interval ϱ of a given length if Π is finitely materialisable for $\mathcal{D}$. In this section, we show that interval ϱ can be further refined if we additionally know that Π is finitely materialisable for *all* bounded datasets, and not just for $\mathcal{D}$.

We start by showing a technical lemma stating that, if a dataset $\mathcal{D}'$ entails all atoms in the grounding $\mathsf{ground}(\Pi, \mathcal{D})$ over a sufficiently long interval, then the facts entailed by $\mathcal{D}'$ and Π outside $[t^{\min}_{\mathcal{D}'}, t^{\max}_{\mathcal{D}'}]$ do not depend on the identity of constants; that is, such a fact of the form $P(\mathbf{s})@\varrho$ is entailed if and only if so is every other fact of the form $P(\mathbf{s}')@\varrho$.

Lemma 25. *For a program Π, a dataset $\mathcal{D}$, and two time points t_1 and t_2 on the $(\Pi, \mathcal{D})$-ruler such that $t_2 - t_1 \geq \mathsf{depth}(\Pi)$, we define a dataset*

$$\mathcal{D}' = \{M'@\varrho' \mid M' \text{ is a relational atom mentioned in } \mathsf{ground}(\Pi, \mathcal{D}) \text{ and } \varrho' \text{ is a } (\Pi, \mathcal{D})\text{-interval contained in } [t_1, t_2]\}.$$

The following two statements hold. If $(\Pi, \mathcal{D}') \models P(\mathbf{s})@t$, for some relational fact $P(\mathbf{s})@t$ with $t > t_2$, then $(\Pi, \mathcal{D}') \models P(\mathbf{s}')@[t_1, t]$ for any tuple $\mathbf{s}'$ (with arity matching $\mathbf{s}$) of constants from Π or $\mathcal{D}$. Similarly, if $t < t_1$, then $(\Pi, \mathcal{D}') \models P(\mathbf{s}')@[t, t_2]$.

Proof. We focus on proving the first statement from the lemma, as the second is symmetric. To this end, we show, inductively on ordinals α, that, if $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s})@t$ and $t > t_2$, then $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s}')@[t_1, t]$, for each tuple $\mathbf{s}'$ (with arity matching $\mathbf{s}$) of constants from Π or $\mathcal{D}$.

The base case holds vacuously, as $T_\Pi^0(\mathfrak{I}_{\mathcal{D}'}) = \mathfrak{I}_{\mathcal{D}'}$ and $t > t_2$ imply that $(\Pi, \mathcal{D}') \not\models P(\mathbf{s})@t$.

In the inductive step for a successor ordinal α, we assume that $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s})@t$. By Lemma 15, $T_\Pi(\mathfrak{I}') \models P(\mathbf{s})@t$, where $\mathfrak{I}'$ is the projection of $T_\Pi^{\alpha-1}(\mathfrak{I}_{\mathcal{D}'})$ over the interval $[t - \mathsf{depth}(\Pi), t + \mathsf{depth}(\Pi)]$. We need to show that $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s}')@t'$, for each $t' \in [t_1, t]$. If $t' \in [t_1, t_2]$, then $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s}')@t'$, by the construction of $\mathcal{D}'$. Assume now that $t' \in (t_2, t]$, let $q = t - t'$, and let $\mathfrak{I}'_{-q}$ be obtained from $\mathfrak{I}'$ by shifting all its facts by q to the left (that is, $\mathfrak{I}'_{-q}$ satisfies a relational fact $M@t''-q$ whenever $\mathfrak{I}'$ satisfies $M@t''$).

We first argue that $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s})@t'$, and then show that $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s}')@t'$. Since $T_\Pi(\mathfrak{I}') \models P(\mathbf{s})@t$, we obtain that $T_\Pi(\mathfrak{I}'_{-q}) \models P(\mathbf{s})@t-q$. Therefore, to prove that we have $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s})@t'$, it is sufficient to show that $\mathfrak{I}'_{-q} \subseteq T_\Pi^{\alpha-1}(\mathfrak{I}_{\mathcal{D}'})$. Indeed, assume that $\mathfrak{I}'_{-q} \models M@t''$, for some relational fact $M@t''$. Hence, $\mathfrak{I}' \models M@t''+q$, and therefore, $T_\Pi^{\alpha-1}(\mathfrak{I}_{\mathcal{D}'}) \models M@t''+q$. By the definition of $\mathfrak{I}'_{-q}$ we know that $t'' > t_1$. If $t''+q \leq t_2$, then, by the definition of $\mathcal{D}'$, $\mathfrak{I}_{\mathcal{D}'} \models M@[t_1, t''+q]$, and thus, $T_\Pi^{\alpha-1}(\mathfrak{I}_{\mathcal{D}'}) \models M@[t_1, t''+q]$. Otherwise, if $t'' + q > t_2$, we use the inductive assumption to obtain $T_\Pi^{\alpha-1}(\mathfrak{I}_{\mathcal{D}'}) \models M@[t_1, t''+q]$.

It remains to show that $t'' \in [t_1, t''+q]$, more specifically, that $t'' \geq t_1$. For this, we observe that all the relational facts that are satisfied in $\mathfrak{I}'_{-q}$ (in particular $M@t''$) are over

time points contained in the interval $[t' - \mathsf{depth}(\Pi), t' + \mathsf{depth}(\Pi)]$, and so, $t'' \geq t' - \mathsf{depth}(\Pi)$. This, by the fact that $t' > t_2$ and $t_2 - t_1 \geq \mathsf{depth}(\Pi)$, implies that $t'' \geq t_1$.

Now, by the inductive assumption, if $\mathfrak{I}'$ satisfies some relational fact, then, by replacing the constants in this fact with any constants mentioned in Π and $\mathcal{D}$, we obtain a fact that is also satisfied in $\mathfrak{I}'$. Thus, $T_\Pi(\mathfrak{I}') \models P(\mathbf{s})@t$ implies $T_\Pi(\mathfrak{I}') \models P(\mathbf{s}')@t$, and so, it implies also $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s}')@t'$.

In the inductive step for a limit ordinal α, we assume that $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s})@t$. Recall that $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) = \bigcup_{\beta<\alpha} T_\Pi^\beta(\mathfrak{I}_{\mathcal{D}'})$, so $T_\Pi^\beta(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s})@t$, for some $\beta < \alpha$. Thus, by the inductive assumption, $T_\Pi^\beta(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s}')@[t_1, t]$, so $T_\Pi^\alpha(\mathfrak{I}_{\mathcal{D}'}) \models P(\mathbf{s}')@[t_1, t]$. This completes the proof of the first statement in the lemma, while the second statement is proved analogously. □

Lemma 25 can now be exploited to identify the relevant—and optimal—segment of the timeline containing all facts entailed by a bounded program Π and dataset $\mathcal{D}$, provided that Π is finitely materialisable for the class of all bounded datasets.

Theorem 26. *Let Π be a bounded program which is finitely materialisable for all bounded datasets, and let $\mathcal{D}$ be a bounded dataset. For each relational fact $M@t$ entailed by Π and $\mathcal{D}$, we have $t \in [t_\mathcal{D}^{\min} - \mathsf{offset}(\Pi), t_\mathcal{D}^{\max} + \mathsf{offset}(\Pi)]$, where $\mathsf{offset}(\Pi) = (\mathsf{pred}(\Pi) - 1) \cdot \mathsf{depth}(\Pi)$.*

Proof. Assume that $(\Pi, \mathcal{D}) \models M@t$ and (without loss of generality) that all predicates in $\mathcal{D}$ are mentioned in Π. We need to show that $t \leq t_\mathcal{D}^{\max} + \mathsf{offset}(\Pi)$ and $t \geq t_\mathcal{D}^{\min} - \mathsf{offset}(\Pi)$; we will focus on showing the first inequality, as the second one has an analogous proof. To this end let $\mathcal{D}'$ be the following dataset, where $t_L = \min\{t_\mathcal{D}^{\min}, t_\mathcal{D}^{\max} - \mathsf{depth}(\Pi)\}$:

$$\mathcal{D}' = \{M'@\varrho' \mid M' \text{ is a relational atom mentioned in } \mathsf{ground}(\Pi, \mathcal{D}) \text{ and } \varrho' \text{ is a } (\Pi, \mathcal{D})\text{-interval contained in } [t_L, t_\mathcal{D}^{\max}]\}.$$

Clearly, $t_L \leq t_\mathcal{D}^{\min}$, and so, $\mathcal{D}'$ entails $\mathcal{D}$. Thus, it suffices to show that $(\Pi, \mathcal{D}') \models M'@t'$ implies $t' \leq t_{\mathcal{D}'}^{\max} + \mathsf{offset}(\Pi)$, for each relational fact $M'@t'$. Notice that $t_{\mathcal{D}'}^{\max} = t_\mathcal{D}^{\max}$, so we will confine ourselves to using the latter symbol. For this, we partition the timeline into an infinite sequence $\ldots, \varrho_{-1}, \varrho_0, \varrho_1, \ldots$ of consecutive $(\Pi, \mathcal{D}')$-intervals, where ϱ_0 is the punctual interval containing $t_\mathcal{D}^{\max}$. Moreover, we define a sequence $\ldots, \mathcal{R}_{-1}, \mathcal{R}_0, \mathcal{R}_1, \ldots$ such that $\mathcal{R}_i$ is the set of relational atoms entailed by Π and $\mathcal{D}'$ over the interval ϱ_i. Let d be the number of $(\Pi, \mathcal{D}')$-intervals contained in $(t_\mathcal{D}^{\max}, t_\mathcal{D}^{\max} + \mathsf{depth}(\Pi)]$, which is even since $(t_\mathcal{D}^{\max}, t_\mathcal{D}^{\max} + \mathsf{depth}(\Pi)]$ contains as many open as punctual $(\Pi, \mathcal{D}')$-intervals. By definition, $\varrho_{(\mathsf{pred}(\Pi)-1)\cdot d}$ is the punctual interval containing the time point $t_\mathcal{D}^{\max} + \mathsf{offset}(\Pi)$. Thus, we need to show that $\mathcal{R}_i$ is empty, for all $i > (\mathsf{pred}(\Pi) - 1) \cdot d$. To this end, we first prove the following statements:

1. It holds that $\mathcal{R}_0 \supseteq \mathcal{R}_1 \supseteq \ldots$.
2. If $\mathcal{R}_i = \mathcal{R}_{i+1} = \cdots = \mathcal{R}_j \neq \emptyset$, for some odd $i > -d$ and $j > i$, then $j < i + d$.
3. If $\mathcal{R}_i = \mathcal{R}_{i+1} = \cdots = \mathcal{R}_j \neq \emptyset$, for some even $i \geq -d$ and $j > i$, then $j < i + d + 1$.
4. There are at most $\mathsf{pred}(\Pi) - 1$ non-empty distinct sets among $\mathcal{R}_1, \mathcal{R}_2, \ldots$.

To prove Statement 1, observe that $t^{\min}_{\mathcal{D}'} = t_L$ and $t^{\max}_{\mathcal{D}} - t^{\min}_{\mathcal{D}'} \geq \mathsf{depth}(\Pi)$. We can thus apply Lemma 25 to $\mathcal{D}'$, which implies that $\mathcal{R}_0 \supseteq \mathcal{R}_1 \supseteq \dots$.

For Statement 2, suppose towards a contradiction that there exists an odd number $i > -d$ such that $\mathcal{R}_i = \dots = \mathcal{R}_{i+d}$. Let $\varrho = \varrho_i \cup \dots \cup \varrho_{i+d}$. Since $\varrho_0 = [t^{\max}_{\mathcal{D}}, t^{\max}_{\mathcal{D}}]$ is a punctual interval, every interval ϱ_k with even k is also punctual, whereas each ϱ_k with odd k is open (on both ends). Hence, as i is odd and d is even, both ϱ_i and ϱ_{i+d} are open. Consequently, ϱ is an open interval of length greater than $\mathsf{depth}(\Pi)$. Moreover, as $0 < i + d$, we have $\varrho_0^+ < \varrho_{i+d}^+$, and so, $t^{\max}_{\mathcal{D}} < \varrho^+$. Therefore, there exist two closed intervals ϱ_1 and ϱ_2 contained in ϱ' and of length $\mathsf{depth}(\Pi)$, such that $t^{\max}_{\mathcal{D}} \leq \varrho_1^+ < \varrho_2^+$. Moreover, Π and $\mathcal{D}'$ entail the same non-empty set of relational atoms in all time points belonging to ϱ_1 and ϱ_2, because both of these intervals are contained in $\varrho = \varrho_i \cup \dots \cup \varrho_{i+d}$ and $\mathcal{R}_i = \dots = \mathcal{R}_{i+d} \neq \emptyset$. Thus, by Statement 1 from Lemma 18, Π and $\mathcal{D}'$ entail infinitely many relational facts over $(\Pi, \mathcal{D}')$-intervals. Consequently, Π is not finitely materialisable for $\mathcal{D}'$, which raises a contradiction. Statement 3 is proved analogously.

We now show Statement 4. By Lemma 25, if $P(\mathbf{s}) \in \mathcal{R}_i$, then $\mathcal{R}_i$ contains also all atoms $P(\mathbf{s}')$, where $\mathbf{s}'$ is any tuple of constants from Π or $\mathcal{D}$ of the same arity as $\mathbf{s}$. Hence, there are at most as many non-empty distinct sets among $\mathcal{R}_0, \mathcal{R}_1, \dots$ as the number $\mathsf{pred}(\Pi)$ of predicates in Π. By Statement 1, it remains to show that $\mathcal{R}_0 \neq \mathcal{R}_1$. Suppose towards a contradiction that $\mathcal{R}_0 = \mathcal{R}_1$. Then, by the definition of $\mathcal{D}'$, we obtain that $\mathcal{R}_{-d} = \dots = \mathcal{R}_0 = \mathcal{R}_1$. Since $-d$ is even, this contradicts Statement 3.

We next exploit Statements 1–4 to show that $\mathcal{R}_i$ is empty, for all $i > (\mathsf{pred}(\Pi) - 1) \cdot d$. For this, we divide the sequence $\mathcal{R}_1, \mathcal{R}_2, \dots$ into the set $\mathfrak{R}$ of maximal subsequences which mention, repeatedly, only one non-empty set. By Statements 1 and 4, $\mathfrak{R}$ has at most $\mathsf{pred}(\Pi) - 1$ sequences. Hence, it remains to show that, on average, each of the sequences in $\mathfrak{R}$ has at most d elements. By Statements 2 and 3, each sequence in $\mathfrak{R}$ has at most $d + 1$ elements, so it suffices to show that each sequence in $\mathfrak{R}$ of length $d + 1$ can be paired with a distinct sequence in $\mathfrak{R}$, of length at most $d - 1$. To this end, assume that $\mathcal{R}_i, \dots, \mathcal{R}_{i+d}$, for some $i \geq 1$, is a sequence of length $d + 1$ in $\mathfrak{R}$. By Statement 2, i needs to be an even number. Now, let $j < i$ be the greatest integer such that $\mathcal{R}_{j-d} = \dots = \mathcal{R}_j$, so $j \geq 0$. We claim that $\mathcal{R}_{j+1}, \dots, \mathcal{R}_{i-1}$ must contain a sequence of length at most $d - 1$ belonging to $\mathfrak{R}$. Towards a contradiction suppose that it is not the case, so $\mathcal{R}_{j+1}, \dots, \mathcal{R}_{i-1}$ can be partitioned into the sequences

$$\mathcal{R}_{j+1}, \dots, \mathcal{R}_{j+d}, \qquad \mathcal{R}_{j+d+1}, \dots, \mathcal{R}_{j+2d}, \qquad \dots, \qquad \mathcal{R}_{i-d}, \dots, \mathcal{R}_{i-1},$$

all of which belong to $\mathfrak{R}$ and are of length d. Since j and d are even, we obtain that the indices of the last elements of these sequences, namely $j + d$, $j + 2d$, $\dots$, $i - 1$, are even numbers. Hence, i is odd, which raises a contradiction. □

The following example shows that the bound from Theorem 26 on the length of the relevant segment of the timeline is optimal.

Example 27. *Consider a dataset $\mathcal{D} = \{P_0@0\}$ and a program Π_n, for any $n \in \mathbb{N}$, consisting of the following rules:*

$$\begin{array}{lllll} \boxplus_1 P_1 \leftarrow P_0, & \boxplus_1 P_2 \leftarrow P_1, & \boxplus_1 P_3 \leftarrow P_2, & \dots, & \boxplus_1 P_n \leftarrow P_{n-1}, \\ \boxminus_1 P_1 \leftarrow P_0, & \boxminus_1 P_2 \leftarrow P_1, & \boxminus_1 P_3 \leftarrow P_2, & \dots, & \boxminus_1 P_n \leftarrow P_{n-1}. \end{array}$$

Algorithm 4: Checking finite materialisability for all bounded datasets

Input: A bounded program Π

Output: A boolean value true if Π is finitely materialisable for all bounded datasets, false otherwise

1 $\mathcal{D}_{\mathsf{new}} \coloneqq \mathcal{D}_\Pi$;
2 $\varrho \coloneqq [t^{\min}_{\mathcal{D}_\Pi} - \mathsf{offset}(\Pi), t^{\max}_{\mathcal{D}_\Pi} + \mathsf{offset}(\Pi)]$;
3 **repeat**
4 $\quad \mathcal{D}_{\mathsf{old}} \coloneqq \mathcal{D}_{\mathsf{new}}$;
5 $\quad \mathcal{D}_{\mathsf{new}} \coloneqq \mathsf{ApplyRules}(\Pi, \mathcal{D}_{\mathsf{old}})$;
6 $\quad$ **if** *there is* $M@\varrho' \in \mathcal{D}_{\mathsf{new}}$ *with* $\varrho' \not\subseteq \varrho$ **then return** false;
7 **until** $\mathfrak{I}_{\mathcal{D}_{\mathsf{old}}} = \mathfrak{I}_{\mathcal{D}_{\mathsf{new}}}$;
8 **return** true;

Since Π_n is non-recursive, by Proposition 13, it is finitely materialisable for any dataset. Thus, by Theorem 26, Π_n and $\mathcal{D}$ entail facts only in $\big[t^{\min}_{\mathcal{D}} - \mathsf{offset}(\Pi_n), t^{\max}_{\mathcal{D}} + \mathsf{offset}(\Pi_n)\big]$. We will show that they entail facts both in the left and in the right endpoints of this range.

Indeed, we start by observing that $\mathsf{pred}(\Pi_n) = n + 1$ and $\mathsf{depth}(\Pi_n) = 1$, so $\mathsf{offset}(\Pi) = n$. Thus, $[-n, n] = \big[t^{\min}_{\mathcal{D}} - \mathsf{offset}(\Pi_n), t^{\max}_{\mathcal{D}} + \mathsf{offset}(\Pi_n)\big]$, as $t^{\min}_{\mathcal{D}} = t^{\max}_{\mathcal{D}} = 0$. Moreover, Π_n and $\mathcal{D}$ entail facts both at $-n$ and n, namely they entail $P_n@{-n}$ and $P_n@n$.

5.3 Checking Data-independent Finite Materialisability

The results in Theorems 24 and 26 suggest that a variant of Algorithm 3 can be used for deciding finite materialisability in the data-independent setting. In particular, we let Algorithm 4 be obtained from Algorithm 3 by dispensing with the input dataset $\mathcal{D}$, replacing $\mathcal{D}$ in Lines 1 and 2 with the critical dataset $\mathcal{D}_\Pi$ as suggested by Theorem 24, and replacing $\mathsf{offset}(\Pi, \mathcal{D})$ in Line 2 with the more refined bound $\mathsf{offset}(\Pi)$ from Theorem 26.

The correctness of Algorithm 4 follows directly from Theorems 24 and 26. As shown next, however, its running time is much more favourable than that of Algorithm 3.

Theorem 28. *Algorithm 4 returns true if the input program Π is finitely materialisable for the class of all bounded datasets, and it returns false otherwise. Moreover, Algorithm 4 runs in exponential time in the size of Π.*

Proof. The correctness of the algorithm is an immediate consequence of Theorems 24 and 26. We next show that the algorithm runs in singly exponential time. As in the analysis of Algorithm 3 in the proof of Theorem 20, we observe that the number of iterations of the main loop in Algorithm 4 is bounded by the maximal number of facts in $\mathcal{D}_{\mathsf{old}}$, which is $A \cdot C$, for A the number of atoms in $\mathsf{ground}(\Pi, \mathcal{D}_\Pi)$ and C the number of $(\Pi, \mathcal{D}_\Pi)$-intervals contained in ϱ. As A is clearly exponential in the size of Π, we next show that C is also exponential. Note that $\varrho = [-\mathsf{offset}(\Pi), \mathsf{depth}(\Pi) + \mathsf{offset}(\Pi)]$, where $\mathsf{offset}(\Pi) = (\mathsf{pred}(\Pi) - 1) \cdot \mathsf{depth}(\Pi)$. Hence, the length of ϱ is $(2 \cdot \mathsf{pred}(\Pi) - 1) \cdot \mathsf{depth}(\Pi)$. Clearly, $\mathsf{pred}(\Pi)$ is polynomial and the number of $(\Pi, \mathcal{D}_\Pi)$-intervals contained in a closed interval of length $\mathsf{depth}(\Pi)$ is $2 \cdot \frac{\mathsf{depth}(\Pi)}{\mathsf{div}(\Pi)} + 1$, which is exponentially large. Thus, C is exponential, and so is the number $A \cdot C$ of loop iterations. Moreover, each iteration of the loop is feasible in exponential time; indeed the

main part of each iteration consists of running Algorithm 1, whose main loop is traversed at most as many times as there are rules in $\mathsf{ground}(\Pi, \mathcal{D}_{\mathsf{old}})$, which is an exponentially large number. Thus, Algorithm 4 runs in exponential time. □

Theorem 28 shows that checking finite materialisability for all bounded datasets is no harder than standard reasoning in vanilla (non-temporal) Datalog. The following theorem shows that this upper bound is tight.

Theorem 29. *Checking whether a bounded program is finitely materialisable for the class of bounded datasets is* ExpTime*-complete.*

Proof. The upper bound is provided by Theorem 28. To show the matching lower bound we reduce fact entailment in Datalog, which is ExpTime-hard even if $\bot$ is disallowed in rule heads (Dantsin et al., 2001). Assume that we want to check if a Datalog program Π consisting of a set Π_F of facts and a set Π_R of non-fact rules—that do not mention $\bot$ in heads—entail a fact M. Note that, syntactically, Π_R is a DatalogMTL program and any Datalog fact is a DatalogMTL relational atom. Hence, we can construct a bounded DatalogMTL program $\Pi' = \Pi_R \cup \{M_F \leftarrow \diamondminus_1 M \mid M_F \in \Pi_F\}$. We will show that Π entails M if and only if Π' is not finitely materialisable for some bounded dataset.

First, we assume that Π entails M. We show that Π' is not finitely materialisable for a bounded dataset $\mathcal{D}' = \{M_F@0 \mid M_F \in \Pi_F\}$, namely, that Π' and $\mathcal{D}'$ entail infinitely many facts over $(\Pi', \mathcal{D}')$-intervals. Indeed, we show inductively on $t \in \mathbb{N}$ that $(\Pi', \mathcal{D}') \models M@t$. For the basis, it suffices to observe that since $\Pi_R \cup \Pi_F$ entails (in Datalog) M, by the construction of $\mathcal{D}'$, we obtain that $(\Pi_R, \mathcal{D}') \models M@0$, and so, $(\Pi', \mathcal{D}') \models M@0$. In the inductive step, we assume that $(\Pi', \mathcal{D}') \models M@t$, for some $t \in \mathbb{N}$. Hence, by the rules of the form $M_F \leftarrow \diamondminus_1 M$ in Π', we have $(\Pi', \mathcal{D}') \models M_F@t{+}1$, for each $M_F \in \Pi_F$. Then, by an analogous argument to the one we used to show the induction basis, we obtain $(\Pi', \mathcal{D}') \models M@t{+}1$, as required.

Second, assume that Π does not entail M. By Theorem 24, it suffices to show that Π' and the critical dataset $\mathcal{D}_{\Pi'}$ entail finitely many relational facts over $(\Pi', \mathcal{D}')$-intervals, or equivalently, that $T^{\omega_1}_{\Pi'}(\mathfrak{I}_{\mathcal{D}_{\Pi'}})$ does so. By the definition, $\mathcal{D}_{\Pi'}$ consists of all relational facts $P(\mathbf{s})@[0,1]$ such that P occurs in Π' and $\mathbf{s}$ mentions only constants from Π' and $c_{\Pi'}$. Hence, as Π does not mention metric operators, we obtain that $T_{\Pi}(\mathfrak{I}_{\mathcal{D}_{\Pi'}}) = \mathfrak{I}_{\mathcal{D}_{\Pi'}}$. If, additionally, $M@[0,1] \notin \mathcal{D}_{\Pi'}$ (which happens if M mentions some predicate or constant not occurring in $\mathcal{D}_{\Pi'}$), then rules of the form $M_F \leftarrow \diamondminus_1 M$ in Π' do not apply, so $T_{\Pi'}(\mathfrak{I}_{\mathcal{D}_{\Pi'}}) = \mathfrak{I}_{\mathcal{D}_{\Pi'}}$. Hence $T^{\omega_1}_{\Pi'}(\mathfrak{I}_{\mathcal{D}_{\Pi'}}) = \mathfrak{I}_{\mathcal{D}_{\Pi'}}$, and thus, $T^{\omega_1}_{\Pi'}(\mathfrak{I}_{\mathcal{D}_{\Pi'}})$ entails finitely many relational facts over $(\Pi', \mathcal{D}')$-intervals. Next, we consider the case when $M@[0,1] \in \mathcal{D}_{\Pi'}$. Then, an application of rules of the form $M_F \leftarrow \diamondminus_1 M$ to $\mathcal{D}_{\Pi'}$ derives facts $\mathcal{D}_1 = \{M_F@(1,2] \mid M_F \in \Pi_F\}$, that is, $T_{\Pi'}(\mathfrak{I}_{\mathcal{D}_{\Pi'}}) = \mathfrak{I}_{\mathcal{D}_{\Pi'} \cup \mathcal{D}_1}$. Since Π' propagates facts only toward the future, no further applications of $T_{\Pi'}$ allow us to derive facts over time points belonging to $(-\infty, 1]$, thus $T^{\omega_1}_{\Pi'}(\mathfrak{I}_{\mathcal{D}_{\Pi'}})$ is the union of $\mathfrak{I}_{\mathcal{D}_{\Pi'}}$ and $T^{\omega_1}_{\Pi'}(\mathfrak{I}_{\mathcal{D}_1})$. To finish the proof, it suffices to show that $T^{\omega_1}_{\Pi'}(\mathfrak{I}_{\mathcal{D}_1})$ entails finitely many relational facts over $(\Pi', \mathcal{D}')$-intervals. By the form of $\mathcal{D}_1$, successive applications of $T_{\Pi'}$ to $\mathfrak{I}_{\mathcal{D}_1}$ derive a relational fact mentioning M if and only if Π entails M. By the assumption, the latter is false, so no relational fact mentioning M occurs in $T^{\omega_1}_{\Pi'}(\mathfrak{I}_{\mathcal{D}_1})$. Thus, rules of the form $M_F \leftarrow \diamondminus_1 M$ will not allow us to add new facts when constructing $T^{\omega_1}_{\Pi'}(\mathfrak{I}_{\mathcal{D}_1})$, therefore $T^{\omega_1}_{\Pi'}(\mathfrak{I}_{\mathcal{D}_1}) = T^{\omega_1}_{\Pi}(\mathfrak{I}_{\mathcal{D}_1})$. Finally, since Π is a Datalog program, it is finitely materialisable, and so, $T^{\omega_1}_{\Pi}(\mathfrak{I}_{\mathcal{D}_1})$ entails finitely many facts over $(\Pi', \mathcal{D}')$-intervals. □

6. Sufficient Conditions for Data-independent Finite Materialisability

When we first introduced the notion of finite materialisability in Section 3, we argued that temporal recursion is the main reason why materialisation may not terminate. In particular, we showed that by focusing on non-recursive DatalogMTL programs (and thus by precluding recursion altogether) we can guarantee finite materialisability for all datasets.

In this section, we propose two incomparable fragments of DatalogMTL which allow for a limited form of temporal recursion while at the same time ensuring data-independent finite materialisability.

- In Section 6.1 we introduce *EDB-guarded* programs, where the body of each rule contains at least one atom involving only EDB predicates (as usual in Databases, these are predicates not mentioned in the head of any rule in the program).
- In Section 6.2 we propose *MTL-acyclic* programs, whose definition is based on a generalisation of the acyclicity condition on the dependency graph used to define non-recursive programs.

As we will show, membership in these fragments can be checked efficiently. Furthermore, while EDB-guardedness ensures finite materialisability of bounded programs for the class of bounded datasets, MTL-acyclicity provides guarantees for possibly unbounded programs and the class of *all* datasets.

6.1 EDB-guarded Programs

We next introduce EDB-guarded DatalogMTL programs where, as usual in the Database literature, we say that a predicate is *extensional* (EDB) in a program Π if this predicate does not occur in the head of any rule of Π.

Definition 30. *A metric atom is* extensional *(EDB) in a program* Π*, if all the predicates mentioned in this atom are EDB in* Π*. Moreover,* Π *is EDB-guarded if each rule of* Π *has at least one EDB body atom.*

Example 31. *Consider our program* Π_{ex} *from Example 1. We can observe that predicates NoSympt, Vaccinated, and Infected are EDB; however, predicates Immune and NegTest are not EDB as they occur in the heads of rules. Hence, although Rules* (1)*,* (2) *and* (3) *are EDB-guarded, Rule* (4) *is not, as it does not contain an EDB atom in the body.*

The next lemma shows that an EDB-guarded bounded program Π cannot entail facts that are 'far away' from the data; in particular, all facts must lie within an interval containing the data and extending at most $\mathsf{depth}(\Pi)$ beyond it towards the future and towards the past. This is so because, for a rule to be applicable, the EDB atom(s) in its body must be matched directly to the explicitly given data.

Lemma 32. *Let* Π *be a bounded EDB-guarded program and* $\mathcal{D}$ *a bounded dataset. Then,* $t \in [t_{\mathcal{D}}^{\min} - \mathsf{depth}(\Pi), t_{\mathcal{D}}^{\max} + \mathsf{depth}(\Pi)]$*, for each relational fact* $M@t$ *entailed by* Π *and* $\mathcal{D}$*.*

Proof. Assume that Π and $\mathcal{D}$ entail $M@t$, so there exists the least ordinal α such that $T_{\Pi}^{\alpha}(\mathfrak{I}_{\mathcal{D}}) \models M@t$. If $\alpha = 0$, then $t \in [t_{\mathcal{D}}^{\min}, t_{\mathcal{D}}^{\max}]$, so $t \in [t_{\mathcal{D}}^{\min} - \mathsf{depth}(\Pi), t_{\mathcal{D}}^{\max} + \mathsf{depth}(\Pi)]$.

Next, assume that $\alpha > 0$, so α is a successor ordinal and $T_\Pi^{\alpha-1}(\mathfrak{I}_\mathcal{D}) \not\models M@t$. By Lemma 15, $T_\Pi(\mathfrak{I}') \models M@t$, for $\mathfrak{I}'$ the projection of $T_\Pi^{\alpha-1}(\mathfrak{I}_\mathcal{D})$ over $[t - \mathsf{depth}(\Pi), t + \mathsf{depth}(\Pi)]$. Now, since Π is EDB-guarded, each of its rules has at least one EDB metric atom in the body. Thus, the fact that $T_\Pi(\mathfrak{I}') \models M@t$ yet $\mathfrak{I}' \not\models M@t$, implies that $\mathfrak{I}'$ satisfies at least one relational fact $M'@t'$ with an EDB atom M'. Since M' is EDB, it cannot occur in the head of any rule in $\mathsf{ground}(\Pi, \mathcal{D})$. Thus, by the fact that $T_\Pi^{\alpha-1}(\mathfrak{I}_\mathcal{D}) \models M'@t'$ (as $\mathfrak{I}' \models M'@t'$ and $\mathfrak{I}' \subseteq T_\Pi^{\alpha-1}(\mathfrak{I}_\mathcal{D})$), we obtain that $\mathfrak{I}_\mathcal{D} \models M'@t'$, and so, $t' \in [t^{\min}_\mathcal{D}, t^{\max}_\mathcal{D}]$. Observe that we have $t' \in [t - \mathsf{depth}(\Pi), t + \mathsf{depth}(\Pi)]$, and so, $t \in [t^{\min}_\mathcal{D} - \mathsf{depth}(\Pi), t^{\max}_\mathcal{D} + \mathsf{depth}(\Pi)]$. □

Lemma 32 immediately ensures finite materialisability of EDB-guarded bounded programs for the class of bounded datasets since the identified interval contains finitely many $(\Pi, \mathcal{D})$-intervals.

Corollary 33. *If a bounded program is EDB-guarded, then it is finitely materialisable for the class of bounded datasets.*

We conclude this section with the observation that the EDB-guardedness condition does not ensure finite materialisability for unbounded programs or unbounded datasets.

Example 34. *Consider a program* $\Pi = \{Q \leftarrow P, \quad \boxplus_1 Q \leftarrow \diamondminus_{[0,\infty)} P \wedge Q\}$. *Even though* Π *is EDB-guarded it is not finitely materialisable for bounded datasets, since the program is unbounded. Indeed,* T_Π *is not finitely materialisable for the dataset* $\mathcal{D} = \{P@1\}$*; for every natural number* $k \geq 1$*, the interpretation* $T_\Pi^k(\mathfrak{I}_\mathcal{D})$ *entails a new fact* $Q@k$*, and so,* T_Π *converges for* $\mathcal{D}$ *in* ω *steps.*

Next, we consider a program $\Pi' = \{Q \leftarrow P \wedge R, \quad \boxplus_1 Q \leftarrow P \wedge Q\}$. *Although* Π' *is EDB-guarded and bounded, it does not finitely materialise for some unbounded datasets. In particular,* $T_{\Pi'}$ *does not converge in finitely many steps for the following unbounded dataset* $\mathcal{D}' = \{R@1, P@[1, \infty)\}$. *For every natural number* $k \geq 1$*, the interpretation* $T_{\Pi'}^k(\mathfrak{I}_{\mathcal{D}'})$ *entails a new fact* $Q@k$*, and thus,* $T_{\Pi'}$ *converges for* $\mathcal{D}'$ *in* ω *steps.*

6.2 MTL-acyclic Programs

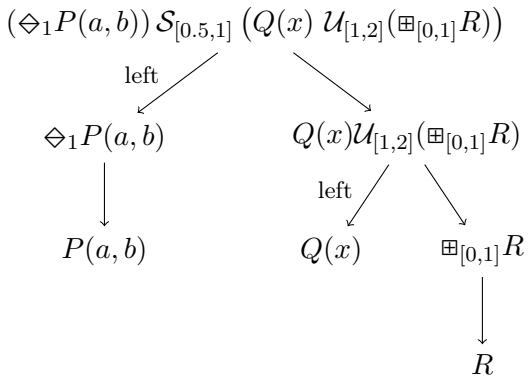
In this section, we propose an acyclicity condition based on an extension of the dependency graph of a DatalogMTL program (see Section 2) to a *metric dependency graph*, where edges representing dependencies between predicates are now labelled with intervals. Our condition precludes certain types of cycles on the metric dependency graph of the program, and will be sufficient to ensure finite materialisability of any (possibly unbounded) program satisfying the condition for the class of *all* datasets. Thus, in contrast to EBD-guardedness which is restricted to the bounded setting, MTL-acyclicity is generally applicable.

The construction of the metric dependency graph for a program Π proceeds in two steps. First, we annotate the *parse tree* T_M of each metric atom M occurring in Π by assigning to each node v corresponding to a sub-atom M' an interval $\mathsf{range}(v, T_M)$ describing how the satisfaction of M can be 'affected' by that of its sub-atom M'. Second, we construct the dependency graph for Π as usual, but annotate all edges with an interval obtained from those computed in the previous step.

The notion of a parse tree for a metric atom is the standard one in logic; in our case it is based on the grammar for metric atoms given in Section 2. For convenience, however, we

will additionally say that an edge (u, v) in a parse tree is a *left edge* if u has two children in the tree and v is the left child. An example is given below for clarity.

Example 35. *Consider a metric atom* $M = (\diamondsuit_1 P(a,b))\, \mathcal{S}_{[0.5,1]}\, \big(Q(x)\, \mathcal{U}_{[1,2]}(\boxplus_{[0,1]} R)\big)$*; the parse tree* T_M *of* M *is as follows, where left edges are indicated explicitly:*



We next show how to assign an interval $\mathsf{range}(v, T_M)$ to each node v representing a sub-atom of M' of M in the parse tree T_M. The intuition is that $\mathsf{range}(v, T_M)$ describes an interval within which satisfaction of M' can affect satisfaction of M. In particular, satisfaction of M at t can be affected by M' satisfied within $t + \mathsf{range}(v, T_M)$.

Definition 36. *The* range*,* $\mathsf{range}(v, T_M)$*, of a vertex* v *in a parse tree* T_M *for a metric atom* M *is the interval* $[0,0] + \varrho_1 + \cdots + \varrho_n$*, where* $v_0, \ldots, v_n$ *is the path from the root* v_0 *of* T_M *to* $v = v_n$*, and* ϱ_i *are defined as follows:*

- $\varrho_i = \varrho$*, if* v_i *is labelled with* $\boxplus_\varrho M'$*,* $\boxplus_\varrho M'$*, or* $M' \mathcal{U}_\varrho M''$ *and* (v_i, v_{i+1}) *is not a left edge,*
- $\varrho_i = (0, \varrho^+)$*, if* v_i *is labelled with* $M' \mathcal{U}_\varrho M''$ *and* (v_i, v_{i+1}) *is a left edge,*
- $\varrho_i = -\varrho$*, if* v_i *is labelled with* $\diamondsuit_\varrho M', \boxminus_\varrho M'$*, or* $M' \mathcal{S}_\varrho M''$ *and* (v_i, v_{i+1}) *is not left edge,*
- $\varrho_i = -(0, \varrho^+)$*, if* v_i *is labelled with* $M' \mathcal{S}_\varrho M''$ *and* (v_i, v_{i+1}) *is a left edge.*

Consider again the metric atom M from Example 35 and its parse tree T_M. For the vertex v labelled with $P(a,b)$ we obtain $\mathsf{range}(v, T_M) = [0,0] - (0,1) - 1 = (-2, 0)$. This means that satisfaction of M at t can be affected by satisfaction of $P(a,b)$ within $(t-2, t)$.

The following lemma formalises the intuition behind Definition 36.

Lemma 37. *Assume that* $T^i_\Pi(\mathfrak{I}_\mathcal{D}) \not\models M@t$ *and* $T^{i+1}_\Pi(\mathfrak{I}_\mathcal{D}) \models M@t$*, for a program* Π*, dataset* $\mathcal{D}$*, relational fact* $M@t$*, and* $i \in \mathbb{N}$*. Then, there exists a relational fact* $M'@t'$ *such that*

1. $T^i_\Pi(\mathfrak{I}_\mathcal{D}) \models M'@t'$*, but* $T^j_\Pi(\mathfrak{I}_\mathcal{D}) \not\models M'@t'$*, for all* $j < i$*, and*
2. *there exists a rule* $r \in \mathsf{ground}(\Pi, \mathcal{D})$ *with head* H *and a body atom* B *such that* $t - t' \in (\mathsf{range}(v, T_H) - \mathsf{range}(v', T_B))$*, where* v *is (the single) leaf of* T_H *labelled with* M *and* v' *is some leaf of* T_B *labelled with* M'*.*

Proof. We first prove Item 1 from the lemma. As $T^i_\Pi(\mathfrak{I}_\mathcal{D}) \not\models M@t$ and $T^{i+1}_\Pi(\mathfrak{I}_\mathcal{D}) \models M@t$, there exists $r \in \mathsf{ground}(\Pi, \mathcal{D})$ whose application at some time point t_r derives $M@t$, namely, the body of r holds at t_r in $T^i_\Pi(\mathfrak{I}_\mathcal{D})$ and the head H of r holding at t_r entails $M@t$, but the body of r does not hold at t_r in any $T^j_\Pi(\mathfrak{I}_\mathcal{D})$, with $j < i$. Hence, some body atom B of r does not hold at t_r in any such $T^j_\Pi(\mathfrak{I}_\mathcal{D})$.

Assume that $B'@t_{B'}$ is a metric fact such that $T^i_\Pi(\mathfrak{I}_\mathcal{D}) \models B'@t_{B'}$ and $T^j_\Pi(\mathfrak{I}_\mathcal{D}) \not\models B'@t_{B'}$, for all $j < i$, where B' is the label of some non-leaf vertex $v_{B'}$ in the parse tree T_B (so B' is a sub-atom of B). We show that there is a metric fact $B''@t_{B''}$ such that $T^i_\Pi(\mathfrak{I}_\mathcal{D}) \models B''@t_{B''}$, and $T^j_\Pi(\mathfrak{I}_\mathcal{D}) \not\models B''@t_{B''}$, for all $j < i$, where B'' is the label of some child $v_{B''}$ of $v_{B'}$ in T_B and $t_{B''} - t_{B'} \in \mathsf{range}(v_{B''}, T_{B'})$.

Indeed, if B' is of the form $\boxplus_\varrho N$ or $\boxplus_\varrho N$, then $T^i_\Pi(\mathfrak{I}_\mathcal{D}) \models B'@t_{B'}$ and $T^j_\Pi(\mathfrak{I}_\mathcal{D}) \not\models B'@t_{B'}$, imply that $T^i_\Pi(\mathfrak{I}_\mathcal{D}) \models N@t_N$ and $T^j_\Pi(\mathfrak{I}_\mathcal{D}) \not\models N@t_N$, for some t_N such that $t_N - t_{B'} \in \varrho$. Observe that N is the label of the single child v_N of B' in T_B, and $\mathsf{range}(v_N, T_{B'}) = \varrho$, so $N@t_N$ witnesses the required $B''@t_{B''}$.

If B' is of the form $N_1 \mathcal{U}_\varrho N_2$, then the argument is similar, however, we obtain that $T^i_\Pi(\mathfrak{I}_\mathcal{D}) \models N@t_N$ and $T^j_\Pi(\mathfrak{I}_\mathcal{D}) \not\models N@t_N$ either for $N = N_2$ and $t_N - t_{B'} \in \varrho$ or for $N = N_1$ and $t_N \in (t_{B'}, t_{B'} + \varrho^+)$. Atoms N_1 and N_2 label the left v_{N_1} and the right v_{N_2} children of $v_{B'}$, respectively, so $\mathsf{range}(v_{N_1}, T_{B'}) = (0, \varrho^+)$ and $\mathsf{range}(v_{N_2}, T_{B'}) = \varrho$. Hence, $N@t_N$ witnesses the required $B''@t_{B''}$.

The remaining cases, with B' of the forms $\diamondminus_\varrho N$, $\boxminus_\varrho N$, and $N_1 \mathcal{U}_\varrho N_2$, are proved analogously.

This implies that there are metric facts $B_1@t_{B_1}, \dots, B_n@t_{B_n}$, all of which hold in $T^i_\Pi(\mathfrak{I}_\mathcal{D})$ but not in any $T^j_\Pi(\mathfrak{I}_\mathcal{D})$ with $j < i$, and such that $B_1@t_{B_1} = B@t_r$, B_n has no metric operators, and $t_{B_n} - t_r \in \mathsf{range}(v', T_B)$, where v' is some leaf of T_B labelled by B_n. Hence, $M'@t' = B_n@t_{B_n}$ satisfies Item 1 in the lemma.

It remains to show that $M'@t'$ satisfies Item 2. As H is a head atom, it can mention only $\boxminus$ and $\boxplus$ among metric operators. Thus, as $H@t_r$ entails $M@t$, by the definition of range, $t - t_r \in \mathsf{range}(v, T_H)$, where v is the single leaf of T_H (labelled by M). Recall that $t' - t_r \in \mathsf{range}(v', T_B)$, so $t - t' \in (\mathsf{range}(v, T_H) - \mathsf{range}(v', T_B))$, as required in Item 2. □

We are ready to define metric dependency graphs.

Definition 38. *The* metric dependency graph G_Π *of a program* Π *is a directed multigraph with edges labelled with intervals, which contains:*

- *a vertex v_P, for each predicate P mentioned in Π,*
- *an edge $(v_P, v_{P'})$ labelled with an interval ϱ, whenever there is a rule r in Π which has a body atom B with a relational atom $P(\mathbf{s})$ and the head H with a relational atom $P'(\mathbf{s}')$ such that $\varrho = \mathsf{range}(v', T_H) - \mathsf{range}(v, T_B)$, where v' is (the single) leaf of T_H labelled with $P'(\mathbf{s}')$ and v is some leaf of T_B labelled with $P(\mathbf{s})$.*

The interval weight of a path *in G_Π is $[0,0] + \varrho_1 + \dots + \varrho_n$, where $\varrho_1, \dots, \varrho_n$ are the intervals labelling the edges on this path.*

Intuitively, graph G_Π has an edge from v_P to $v_{P'}$ labelled with ϱ if there is a rule r in Π such that, whenever some relational atom with predicate P holds at a time point t, then

the application of r *may* derive some relational fact mentioning P' at a time point t' such that $t' - t \in \varrho$. Note also that by removing all interval labels from edges (and then deleting repeated unlabelled edges) from G_Π we obtain the standard dependency graph.

We now define MTL-acyclic programs as those whose dependency graph does not contain certain types of cycles.

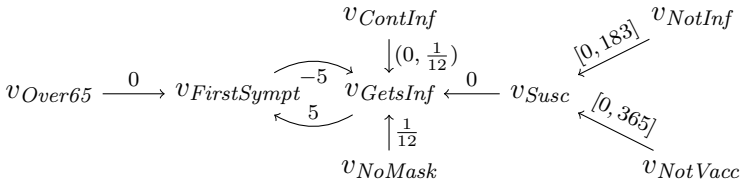
Definition 39. *We say that a program Π is MTL-acyclic if its metric dependency graph G_Π does not contain a cycle with interval weight different from $[0, 0]$.*

Non-recursive DatalogMTL programs, as well as all plain Datalog programs, are trivially MTL-acyclic. Furthermore, programs that exhibit 'safe' temporal recursion are also MTL-acyclic, as illustrated in the following example.

Example 40. *It is believed that individuals who have been neither vaccinated in the last year nor infected for the last half a year are susceptible to COVID-19. Usually, a susceptible individual who takes off their mask and remains in contact with an infected person for 2 hours gets infected. Individuals over 65 develop first symptoms 5 days after getting infected. It is also estimated that for individuals who develop first symptoms of COVID-19, the source of infection should be looked for on the 5th day preceding the current one. A* DatalogMTL *program representing these dependencies consists of the following rules:*

$$\begin{aligned}
\mathit{Susc}(x) &\leftarrow \boxminus_{[0,365]} \mathit{NotVacc}(x) \wedge \boxminus_{[0,183]} \mathit{NotInf}(x),\\
\mathit{GetsInf}(x) &\leftarrow \mathit{ContInf}(x,y) \mathcal{S}_{\frac{1}{12}} \mathit{NoMask}(x) \wedge \mathit{Susc}(x),\\
\mathit{FirstSympt}(x) &\leftarrow \boxminus_5 \mathit{GetsInf}(x) \wedge \mathit{Over65}(x),\\
\boxminus_5 \mathit{GetsInf}(x) &\leftarrow \mathit{FirstSympt}(x).
\end{aligned}$$

The metric dependency graph of this program is given below:



The graph has one simple cycle and its weight is $[0, 0]$, so the program is MTL-acyclic.

The following theorem shows that MTL-acyclicity is a sufficient condition for finite materialisability for all datasets. More precisely, MTL-acyclicity ensures that the materialisation of a (possibly unbounded) program Π for a (possibly unbounded) dataset $\mathcal{D}$ will terminate after a number of rounds of rule applications bounded by the (exponentially large) number of relational facts mentioned in the grounding of Π with respect to $\mathcal{D}$.

Theorem 41. *For every MTL-acyclic program Π and for every dataset $\mathcal{D}$, it holds that $\mathfrak{C}_{\Pi,\mathcal{D}} = T_\Pi^{k-1}(\mathfrak{I}_\mathcal{D})$, where k is the number of relational atoms mentioned in $\mathsf{ground}(\Pi, \mathcal{D})$.*

Proof. Suppose towards a contradiction that $\mathfrak{C}_{\Pi,\mathcal{D}} \neq T_\Pi^{k-1}(\mathfrak{I}_\mathcal{D})$, so $T_\Pi^0(\mathfrak{I}_\mathcal{D}) \neq \ldots \neq T_\Pi^k(\mathfrak{I}_\mathcal{D})$. Therefore, by Lemma 37, there exists a sequence $M_0@t_0, \ldots, M_k@t_k$ of relational facts obtained by subsequently applying, not necessarily distinct, rules $r_0, \ldots, r_{k-1} \in \mathsf{ground}(\Pi, \mathcal{D})$; that is, the following properties hold for each $i \in \{0, \ldots, k\}$:

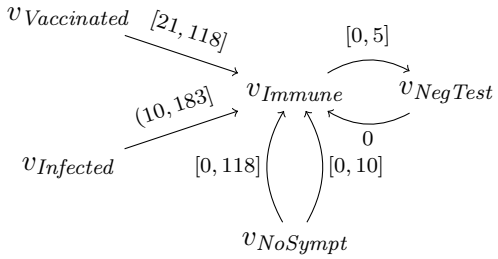
1. $T^i_\Pi(\mathfrak{I}_\mathcal{D}) \models M_i@t_i$, but $T^j_\Pi(\mathfrak{I}_\mathcal{D}) \not\models M_i@t_i$ for all $j < i$, and
2. $t_{i+1} - t_i \in \mathsf{range}(v_{M_{i+1}}, T_{H_i}) - \mathsf{range}(v_{M_i}, T_{B_i})$, for $i < k$, where H_i is the head of r_i, B_i is some body atom of r_i, $v_{M_{i+1}}$ is (the single) leaf of T_{H_i} labelled with M_{i+1}, and v_{M_i} is some leaf of T_{B_i} labelled with M_i.

Since each of $M_0, \dots, M_k$ is mentioned in $\mathsf{ground}(\Pi, \mathcal{D})$ and the number of relational atoms in $\mathsf{ground}(\Pi, \mathcal{D})$ is k, there exist j and ℓ such that $j < \ell \leq k$ and $M_\ell = M_j$. We will show that $t_\ell = t_j$, and thus, $T^j_\Pi(\mathfrak{I}_\mathcal{D}) \models M_\ell@t_\ell$. Since $j < \ell$, it will contradict Item 1 above.

Let Π' be the propositional program obtained from $\mathsf{ground}(\Pi, \mathcal{D})$ by replacing each ground relational atom M with a fresh proposition P_M. Hence, $G_{\Pi'}$ has k vertices. Moreover, by Definition 38, for each $i \in \{j, \dots, \ell - 1\}$ graph $G_{\Pi'}$ has an edge from $v_{P_{M_i}}$ to $v_{P_{M_{i+1}}}$ labelled with $\mathsf{range}(v_{M_{i+1}}, T_{H_i}) - \mathsf{range}(v_{M_i}, T_{B_i})$. Thus, $G_{\Pi'}$ has a path p from $v_{P_{M_j}}$ to $v_{P_{M_\ell}}$, of interval weight $\sum_{i=j}^{\ell-1} \big(\mathsf{range}(v_{M_{i+1}}, T_{H_i}) - \mathsf{range}(v_{M_i}, T_{B_i})\big)$. As $M_\ell = M_j$, this path is a cycle. Moreover, we can show that the interval weight of p is $[0, 0]$. Indeed, recall that $v_{P_{M_j}}, \dots, v_{P_{M_\ell}}$ are the vertices of p and assume towards a contradiction that the interval weight of p is not $[0, 0]$. Let $Q_j, \dots, Q_\ell$ be the predicates mentioned in $M_j, \dots, M_\ell$, respectively. For each $i \in \{j, \dots, \ell - 1\}$, if $G_{\Pi'}$ has an edge from $v_{P_{M_i}}$ to $v_{P_{M_{i+1}}}$ labelled with ϱ, then G_Π has an edge from v_{Q_i} to $v_{Q_{i+1}}$ labelled with the same interval ϱ. Hence, G_Π has a cycle $v_{Q_j}, \dots, v_{Q_\ell}$ of interval weight different than $[0, 0]$, raising a contradiction, as Π is MTL-acyclic. Thus, we have shown that $\sum_{i=j}^{\ell-1} \big(\mathsf{range}(v_{M_{i+1}}, T_{H_i}) - \mathsf{range}(v_{M_i}, T_{B_i})\big) = [0, 0]$. However, by Item 2, $t_\ell - t_j \in \sum_{i=j}^{\ell-1} \big(\mathsf{range}(v_{M_{i+1}}, T_{H_i}) - \mathsf{range}(v_{M_i}, T_{B_i})\big)$, so $t_\ell = t_j$. □

Corollary 42. *MTL-acyclic programs are finitely materialisable for the class of all datasets.*

Example 43. *Observe that the program Π_{ex} from Example 1 is not MTL-acyclic, since its metric dependency graph has a cycle of non-0 weight:*



We observe that even though Π_{ex} is not MTL-acyclic, it is finitely materialisable for the class of all datasets (the immediate consequence operator converges in at most 2 steps). This shows that MTL-acyclicity is not a necessary condition for finite materialisability.

When restricted to bounded programs and bounded datasets, MTL acyclicity also refines the relevant segment of the timeline over which facts can be entailed. In particular, the interval bounds established in Theorem 26 can be further reduced as shown next.

Theorem 44. *Let Π be a bounded MTL-acyclic program and $\mathcal{D}$ a bounded dataset. Then, $t \in [t^{\min}_\mathcal{D} - \mathsf{maxpath}(\Pi), t^{\max}_\mathcal{D} + \mathsf{maxpath}(\Pi)]$ for each relational fact $M@t$ entailed by Π and $\mathcal{D}$, where $\mathsf{maxpath}(\Pi)$ is the maximal absolute value of an endpoint of interval weights in simple paths of G_Π.*

Proof. Assume that $M@t$ is a relational fact entailed by Π and $\mathcal{D}$. Then, by Theorem 41, there exists $\ell \leq k-1$, where k is the number of relational atoms occurring in $\mathsf{ground}(\Pi, \mathcal{D})$, such that $T_\Pi^\ell(\mathfrak{I}_\mathcal{D}) \models M@t$, but $T_\Pi^j(\mathfrak{I}_\mathcal{D}) \not\models M@t$, for all $j < \ell$. Hence, by Lemma 37, there is a sequence $M_0@t_0, \ldots, M_\ell@t_\ell$ of relational facts—with $M_\ell@t_\ell = M@t$—obtained by subsequently applying, not necessarily distinct, rules $r_0, \ldots, r_{\ell-1}$ belonging to $\mathsf{ground}(\Pi, \mathcal{D})$; that is, such that Items 1 and 2 from the proof of Theorem 41 hold for each $i \in \{0, \ldots, \ell\}$.

It remains to show that $t_\ell \in [t_0 - \mathsf{maxpath}(\Pi), t_0 + \mathsf{maxpath}(\Pi)]$; indeed, this will imply that $t \in [t_\mathcal{D}^{\min} - \mathsf{maxpath}(\Pi), t_\mathcal{D}^{\max} + \mathsf{maxpath}(\Pi)]$, as $t_\ell = t$ and $t_0 \in [t_\mathcal{D}^{\min}, t_\mathcal{D}^{\max}]$. To this end, let $\varrho = \sum_{i=0}^{\ell-1} \big(\mathsf{range}(v_{M_{i+1}}, T_{H_i}) - \mathsf{range}(v_{M_i}, T_{B_i})\big)$, where H_i and B_i are as stated in Item 2 from the proof of Theorem 41. Hence, by Item 2, $t_{i+1} - t_i \in \mathsf{range}(v_{M_{i+1}}, T_{H_i}) - \mathsf{range}(v_{M_i}, T_{B_i})$, for each $i \in \{0, \ldots, \ell-1\}$, and therefore, $t_\ell - t_0 \in \varrho$. Let $P_1, \ldots, P_\ell$ be the predicates occurring in $M_1, \ldots, M_\ell$, respectively. By the definition of metric dependency graph, for each $i \in \{0, \ldots, \ell-1\}$, there exists in G_Π an edge from v_{P_i} to $v_{P_{i+1}}$ with interval weight $\mathsf{range}(v_{M_{i+1}}, T_{H_i}) - \mathsf{range}(v_{M_i}, T_{B_i})$. Consequently, the interval weight of the path $p = v_{P_0}, \ldots, v_{P_\ell}$ is equal to ϱ. Now, let p' be any simple path obtained from p by removing all cycles. Since Π is MTL-acyclic, each deleted cycle has interval weight $[0,0]$, so the interval weight of p' is the same as of p, that is, ϱ. Since p' is a simple path, by the definition of $\mathsf{maxpath}(\Pi)$, we get that $-\mathsf{maxpath}(\Pi) \leq \varrho^- $ and $\varrho^+ \leq \mathsf{maxpath}(\Pi)$. Thus, $t_\ell - t_0 \in \varrho$ implies $t_\ell \in [t_0 - \mathsf{maxpath}(\Pi), t_0 + \mathsf{maxpath}(\Pi)]$. □

We conclude this section by showing that MTL acyclicity can be checked efficiently.

Theorem 45. *Checking whether a program is MTL-acyclic is* NL*-complete.*

Proof. For the lower bound we provide a reduction from reachability in directed graphs. To check if a vertex t is reachable from a vertex s in a directed graph G, we construct a bounded program Π mentioning propositions P_v for all vertices v in G, such that, if there is an edge in G from v to v', then Π has a rule $P_{v'} \leftarrow P_v$. Moreover, Π has an additional rule $P_s \leftarrow \diamondsuit_1 P_t$. Then, t is reachable from s in G if and only if there is a simple cycle in the metric dependency graph G_Π of interval weight different from $[0,0]$, which, in turn, holds if and only if Π is not MTL-acyclic. Since NL = coNL, the required NL-hardness follows.

For the upper bound, we provide a logarithmic space reduction from the complement of our problem to the problem of checking whether a simple cycle of non-0 weight exists in a directed graph where edges are weighted with (potentially negative) rational numbers given in binary or with $-\infty$ or ∞ (we assume that $\infty - \infty = \infty$). The latter is feasible in NL by guessing the required non-0 weight cycle edge by edge. Note that we cannot keep in logarithmic memory the weight of the so-far guessed path, due to the binary representation of edge labels. Instead, we can keep in memory this value modulo a previously guessed prime number which does not divide any denominator of a label occurring in the cycle. This result is discussed in the Appendix.

For the reduction we construct a graph G by modifying the metric dependency graph G_Π as follows. For each vertex v in G_Π, the graph G has a pair of vertices v^- and v^+. Moreover, for each edge e in G_Π from v to u of interval weight ϱ, the graph G has an edge e^- from v^- to u^- of weight ϱ^- and an edge e^+ from v^+ to u^+ and of weight ϱ^+. We will show that Π is not MTL-acyclic—that is, G_Π has a simple cycle of interval weight different from $[0,0]$—if and only if G has a simple cycle whose weight is different from 0.

For the forward direction, assume that G_Π has a simple cycle whose edges $e_1, \dots, e_n$ have interval weights $\varrho_1, \dots, \varrho_n$, respectively, and $\varrho = \varrho_1 + \dots + \varrho_n$ is not $[0, 0]$. Hence, $\varrho^- \neq 0$ or $\varrho^+ \neq 0$. If $\varrho^- \neq 0$, then $\varrho_1^- + \dots + \varrho_n^- \neq 0$, and since edges $e_1^-, \dots, e_n^-$ in G have weights $\varrho_1^-, \dots, \varrho_n^-$, respectively, they form a simple cycle of weight $\varrho^- \neq 0$. If $\varrho^+ \neq 0$, we obtain, analogously, that $e_1^+, \dots, e_n^+$ constitute a simple cycle in G of weight $\varrho^+ \neq 0$.

For the opposite direction, assume that G has a simple cycle whose edges $e_1^{x_1}, \dots, e_n^{x_n}$, for $x_i \in \{-, +\}$, have weights $w_1, \dots, w_n$, respectively, and $w_1 + \dots + w_n \neq 0$. By the construction of G, there is a simple cycle $e_1, \dots, e_n$ in G_Π whose edges have some interval weights $\varrho_1, \dots, \varrho_n$. The construction of G implies also that either all $x_1, \dots, x_n$ are $-$, or all of them are $+$. If all $x_1, \dots, x_n$ are $-$, then $w_i = \varrho_i^-$ for each $i \in \{1, \dots, n\}$, and so, $\varrho_1^- + \dots + \varrho_n^- \neq 0$. Similarly, if all $x_1, \dots, x_n$ are $+$, then $\varrho_1^+ + \dots + \varrho_n^+ \neq 0$. Thus, in both cases $\varrho_1 + \dots + \varrho_n \neq [0, 0]$, as required.

Finally, we observe that the reduction is feasible in logarithmic space. In particular, constructing G_Π boils down to adding binary numbers, which is required to compute intervals labelling edges, and is feasible in logarithmic space. The transformation of G_Π to G is also clearly feasible in logarithmic space. □

7. Fact Entailment

We now turn our attention to fact entailment. We first consider bounded programs and datasets and show that reasoning is as hard as for arbitrary DatalogMTL in both combined and data complexity. We then focus on fact entailment over finitely materialisable programs in the data-independent setting, and consider also the particular cases of EDB-guarded programs and MTL-acyclic programs.

7.1 Fact Entailment in Bounded DatalogMTL

Many of our technical results in this paper apply to bounded DatalogMTL programs and datasets; although the complexity of DatalogMTL and its fragments has received significant attention in recent years (Brandt et al., 2018; Wałęga et al., 2019, 2020b), the computational properties of bounded programs have not yet been explored. In this section, we establish the complexity of fact entailment in this fragment and show that reasoning is as hard as in the unrestricted language both in combined and in data complexity. We start by showing the data complexity bounds.

Theorem 46. *Checking whether a bounded program and a bounded dataset entail a bounded fact is* PSpace*-complete for data complexity.*

Proof. The upper bound is immediate, since fact entailment for arbitrary (not necessarily bounded) programs and datasets is PSpace-complete (Wałęga et al., 2019, Theorem 13).

We establish the matching lower bound by employing the reduction from the proof of Theorem 21, where, given a deterministic Turing machine $\mathfrak{M}$ using polynomially many tape cells and a word w, we constructed a bounded program $\Pi_\mathfrak{M}$ and a bounded dataset $\mathcal{D}_w$ such that $\mathfrak{M}$ accepts w if and only if $\Pi_\mathfrak{M}$ is not finitely materialisable for $\mathcal{D}_w$. Now we observe that, by the equivalence of Statements 1–3 in the proof of Theorem 21, if $\mathfrak{M}$ accepts w, then $\Pi_\mathfrak{M}$ and $\mathcal{D}_w$ entail $G@1$. If, on the other hand, $\mathfrak{M}$ does not accept w, then $\Pi_\mathfrak{M}$ and $\mathcal{D}_w$ do not entail $G@t$, for any time point t, and so, they do not entail $G@1$. Thus, we obtain

a reduction to fact entailment, where only the dataset depends on the input word for the machine. Hence, PSpace-hardness for data complexity follows. □

We next establish the combined complexity bounds.

Theorem 47. *Checking whether a bounded program and a bounded dataset entail a bounded fact is* ExpSpace*-complete for combined complexity.*

Proof. The upper bound is inherited from reasoning with arbitrary DatalogMTL programs and datasets (Brandt et al., 2018, Theorem 8 and Proposition 3).

For the lower bound, we modify the reduction from the proof of Theorem 22, where given a deterministic Turing machine $\mathfrak{M}$ with an exponentially long tape and a word w, we constructed a bounded program Π'' and a bounded dataset $\mathcal{D}$ such that $\mathfrak{M}$ halts on w if and only if Π'' and $\mathcal{D}$ entail $H_{q_h,a}@(-\infty,\infty)$, for some tape symbol a. Now, we construct a program Π''' by adding to Π'' rules $\mathit{Halt} \leftarrow H_{q_h,a}$, for all tape symbols a. Thus, $\mathfrak{M}$ halts on w if and only if Π''' and $\mathcal{D}$ entail the fact $\mathit{Halt}@0$. □

7.2 Fact Entailment in Finitely Materialisable Programs

In Section 7.1, we showed that fact entailment for bounded programs and datasets is PSpace-complete in data complexity and ExpSpace-complete for combined complexity, and hence, as hard as for arbitrary DatalogMTL programs and datasets.

We now focus our attention on programs that are finitely materialisable—either semantically or perhaps because they satisfy one of the sufficient conditions in Section 6—and analyse whether fact entailment becomes computationally easier as a result.

We start by providing complexity upper bounds for bounded programs that are finitely materialisable for the class of all bounded datasets. The PSpace upper bound transfers seamlessly from full DatalogMTL. Regarding the combined complexity, we show that it drops from ExpSpace for full DatalogMTL to ExpTime, which is the combined complexity of fact entailment for plain Datalog.

Theorem 48. *Checking whether a bounded program which is finitely materialisable for all bounded datasets and a bounded dataset entail a fact is in* ExpTime *for combined complexity and in* PSpace *for data complexity.*

Proof. The upper bound for data complexity follows trivially from PSpace-completeness of fact entailment for arbitrary DatalogMTL programs and datasets (Wałęga et al., 2019, Theorem 13). To show the ExpTime bound for combined complexity, assume that we want to check if $(\Pi, \mathcal{D}) \models M@\varrho$. To this end, we use Algorithm 2 to construct a dataset $\mathcal{D}'$ representing $\mathfrak{C}_{\Pi,\mathcal{D}}$ and check if $\mathcal{D}' \models M@\varrho$. By Theorem 26, all relational facts entailed by $\mathfrak{C}_{\Pi,\mathcal{D}}$ lie within the interval $[t^{\min}_{\mathcal{D}} - \mathsf{offset}(\Pi), t^{\max}_{\mathcal{D}} + \mathsf{offset}(\Pi)]$, where we recall that $\mathsf{offset}(\Pi) = (\mathsf{pred}(\Pi) - 1) \cdot \mathsf{depth}(\Pi)$, so the fact that $[t^{\min}_{\mathcal{D}}, t^{\max}_{\mathcal{D}}]$, as well as each interval of length $\mathsf{depth}(\Pi)$, has at most exponentially many $(\Pi, \mathcal{D})$-intervals (see the proof of Theorem 20), implies that the same holds for $[t^{\min}_{\mathcal{D}} - \mathsf{offset}(\Pi), t^{\max}_{\mathcal{D}} + \mathsf{offset}(\Pi)]$. This, as we argued in the proof of Theorem 29, implies that Algorithm 2 works in exponential time. □

The matching lower bounds hold already for EDB-guarded programs, as we show next.

Theorem 49. *Checking whether an EDB-guarded bounded program and a bounded dataset entail a fact is* ExpTime*-hard for combined complexity and* PSpace*-hard for data complexity.*

Proof. To show PSpace-hardness for data complexity, we will slightly modify the reduction from the proof of Theorem 21, where, given a deterministic Turing machine $\mathfrak{M}$ using polynomially many tape cells and a word w, we constructed a bounded program $\Pi_{\mathfrak{M}}$ and a bounded dataset $\mathcal{D}_w$ such that $\mathfrak{M}$ accepts w if and only if $\Pi_{\mathfrak{M}}$ is not fintely materialisable for $\mathcal{D}_w$. Now, we use the same $\mathcal{D}_w$, but we construct a new program $\Pi'_{\mathfrak{M}}$ by replacing in $\Pi_{\mathfrak{M}}$ the rule $\boxminus_1 G \leftarrow G$ with $\boxminus_1 G \leftarrow G \wedge \mathit{Tape}$. As a result, all the rules in $\Pi'_{\mathfrak{M}}$ have an EDB-atom *Tape* in bodies, and so, $\Pi_{\mathfrak{M}}$ is EDB-guarded. Moreover, as in the case of unmodified $\Pi_{\mathfrak{M}}$, we obtain that $\mathfrak{M}$ accepts w if and only if $\Pi'_{\mathfrak{M}}$ and $\mathcal{D}_w$ entail $G@1$, which yields PSpace-hardness.

To show the ExpTime lower bound for combined complexity, we reduce fact entailment in plain Datalog, which is ExpTime-complete (Dantsin et al., 2001). Assume that we want to check if a Datalog program Π consisting of a set Π_F of facts and a set Π_R of non-fact rules—that do not mention $\bot$ in heads—entail a fact M. We construct an EDB-guarded bounded DatalogMTL program Π' obtained by adding a fresh (and so, EDB in Π') body atom P to each rule in Π. Moreover, we construct a bounded DatalogMTL dataset $\mathcal{D}' = \{M_F@0 \mid M_F \in \Pi_F\} \cup \{P@0\}$. It is straightforward to check that Π entails M if and only if Π' and $\mathcal{D}'$ entail $M@0$. □

Observe, however, that Theorems 48 and 49 do not provide tight complexity bounds for MTL-acyclic programs, which are incomparable with EDB-guarded programs. We conclude this section by showing that fact entailment over MTL-acyclic programs and arbitrary datasets has exactly the same combined and data complexity as fact entailment in plain Datalog; in particular, this implies tractability in data complexity. This is a surprising result since the class of MTL-acyclic programs significantly extends plain Datalog by allowing all types of metric operators in rules and supporting a limited form of temporal recursion.

Theorem 50. *Checking whether an MTL-acyclic program and a dataset entail a fact is* ExpTime*-complete for combined complexity and* P*-complete for data complexity.*

Proof. For the lower bounds it suffices to observe that each Datalog program is MTL-acyclic, and fact entailment in Datalog is ExpTime-complete in combined and P-complete in data complexity (Dantsin et al., 2001).

For the matching upper bounds we exploit Theorem 41, which states that it suffices to apply the immediate consequence operator k times to obtain the canonical interpretation of an MTL-acyclic program Π and a dataset $\mathcal{D}$, where k is the number of relational atoms mentioned in $\mathsf{ground}(\Pi, \mathcal{D})$. Hence, we use Algorithm 2 on inputs Π and $\mathcal{D}$ to construct a dataset $\mathcal{D}'$ representing the canonical interpretation of Π and $\mathcal{D}$. The algorithm terminates after iterating its main loop at most k times. Each iteration of the loop involves running ApplyRules, that is, Algorithm 1, which, in turn, performs its main loop as many times as there are rules in $\mathsf{ground}(\Pi, \mathcal{D})$. We observe that $\mathsf{ground}(\Pi, \mathcal{D})$ (and so, k) is exponential in combined and polynomial in data complexity. Thus, Algorithm 2 does not exceed the computational resources stated in the theorem. Finally, we use the dataset $\mathcal{D}'$ constructed by Algorithm 2 to check if $\mathcal{D}' \models M@\varrho$. To this end, it suffices to verify if there are facts $M@\varrho_1, \dots, M@\varrho_k$ in $\mathcal{D}'$ such that $\varrho \subseteq (\varrho_1 \cup \dots \cup \varrho_k)$, which is feasible in polynomial time. □

The results in this section provide a full picture of the complexity of reasoning over finitely materialisable programs.

8. Related Work

In this section, we review the relevant literature on temporal reasoning and discuss reasoning problems in Knowledge Representation and Databases related to finite materialisability.

8.1 Languages for Temporal Reasoning

DatalogMTL was first introduced by Brandt et al. (2017) under the continuous semantics and over the rational timeline. The complexity of standard reasoning tasks in DatalogMTL and its fragments was further investigated by Brandt et al. (2018), Wałęga et al. (2019, 2020b), Bellomarini, Nissl, and Sallinger (2021). DatalogMTL has also been studied over the integer timeline (Wałęga et al., 2020a) and under the event-based semantics (Ryzhikov et al., 2019). A language similar to DatalogMTL was first introduced by Brzoska (1998), who also proposed a reasoning algorithm based on solving linear inequalities. DatalogMTL has recently been extended with negation-as-failure under the stable model semantics, first for stratified programs (Tena Cucala et al., 2021) and subsequently for the general case (Wałęga et al., 2021a). MeTeoR (Wang et al., 2022) is the first reasoner to support the full DatalogMTL language by combining materialisation-based and automata-based techniques, and has been successfully applied for solving stream reasoning tasks (Schneider et al., 2022). Earlier systems with DatalogMTL support, such as the Ontop platform (Kalayci et al., 2019), performed reasoning via rewriting into SQL and were restricted to non-recursive programs. DatalogMTL has proved useful for temporal stream reasoning (Wałęga et al., 2019; Schneider et al., 2022), temporal ontology-based data access (Brandt et al., 2017), specification and verification of banking agreements (Nissl & Sallinger, 2022), fact-checking economic claims (Mori et al., 2022), and for describing dance movements (Raheb et al., 2017), among others.

There have been numerous other proposals for extending Datalog with temporal constructs, and we next discuss those that are most relevant to our work. Datalog_{1S} (Chomicki & Imielinski, 1988) extends Datalog by allowing terms of an additional temporal sort and a single successor operator over this sort. Datalog_{1S} is expressively equivalent to TempLog, which augments plain Datalog with linear temporal logic (LTL) operators (Abadi & Manna, 1989; Baudinet, 1989); both of these languages are interpreted over the integer timeline and can be seen as strict subsets of DatalogMTL. A similar language to Datalog_{1S}, called Temporal Datalog, was studied and applied by Ronca, Kaminski, Cuenca Grau, and Horrocks (2022) to stream reasoning. Temporal Datalog is very similar to Datalog_{1S}, but it allows for binary encoding of time points. Another extension of Datalog_{1S}, known as Datalog_{nS}, allows for a limited use of n-ary functions for the temporal sort (Chomicki, 1995). There is a number of other extensions of Datalog with operators from LTL (Artale et al., 2015) and from the Halpern-Shoham logic of intervals (Kontchakov et al., 2016).

Extensions of DatalogMTL with non-monotonic negation are closely related to temporal extensions of answer set programming (ASP) (Aguado et al., 2021). Particularly relevant is a recently introduced extension of ASP with MTL operators (Cabalar et al., 2020). Previously, ASP was also extended with LTL operators, giving rise to temporal equilibrium logic TEL (Cabalar & Pérez Vega, 2007; Diéguez, 2012; Cabalar, 2022), which has been imple-

mented in the Telingo system (Cabalar et al., 2019). Additionally, LARS (Beck et al., 2018) combines ASP and MTL operators for reasoning over data streams. It is worth observing that, unlike DatalogMTL, all these temporal extensions of ASP are interpreted over the integer timeline. *Bidirectional ASP programs* provide another related extension of ASP with functional symbols—they allow for modeling the integer timeline while ensuring decidability of reasoning (Eiter & Simkus, 2009, 2010).

Operators from MTL have also been exploited in temporal extensions of description logics (DLs) (Gutiérrez-Basulto et al., 2016c; Baader et al., 2017; Thost, 2018; Artale & Franconi, 1998). Other temporal DLs rely on LTL operators or interval operators from the Halpern-Shoham logic, among others (Ozaki et al., 2018; Kovtunova, 2017; Artale & Franconi, 2000; Artale et al., 2014; Ryzhikov et al., 2020; Artale et al., 2014, 2015a, 2015b; Gutiérrez-Basulto et al., 2016b).

8.2 Related Problems and Techniques

Our data-dependent and data-independent notions of finite materialisability are respectively related to checking finiteness of the canonical model for a given dataset (known as *weak safety)*, and checking finiteness for all datasets (*strong safety*). These problems were studied by Chomicki and Imielinski (1988) in the context of Datalog_{1S}, who showed that weak safety is in PSpace for data complexity whereas strong safety is in ExpTime. It is worth observing, however, that finite materialisability and finiteness of the canonical model do not coincide in the context of DatalogMTL; in particular, since we study DatalogMTL over the rational timeline, its canonical model may satisfy infinitely many rational facts even if the model is constructed in a finite number of materialisation steps.

Finite materialisability is also related to the termination of *chase* procedures in (non-temporal) Datalog with existential rules (also known as $\text{Datalog}^{\exists}$, $\text{Datalog}^{\pm}$, or tuple generating dependencies) (Fagin et al., 2005; Cuenca Grau et al., 2013; Marnette & Geerts, 2010; Krötzsch et al., 2019; Baget et al., 2014). Deutsch, Nash, and Remmel (2008, Theorems 6 and 7) showed that termination of the core chase is undecidable for a given dataset, whereas Grahne and Onet (2018, Theorem 5.14) proved that *universal termination*—that is, termination for all datasets—is also undecidable. Undecidability of universal termination was established also for the *oblivious* chase (Gogacz & Marcinkowski, 2014, Theorem 1), and even for signatures consisting of only unary and binary predicates (Bednarczyk et al., 2020, Theorem 1.1). In contrast, as we showed in this paper, both data-dependent and data-independent finite materialisability checking are decidable for DatalogMTL.

A standard technique for checking universal termination of a chase is based on reducing the problem to checking termination for a single *critical dataset* (Marnette, 2009; Calautti et al., 2015; Calautti & Pieris, 2021; Bednarczyk et al., 2020). Depending on the type of chase, this technique may require specific modifications (Gogacz et al., 2020; Karimi et al., 2021). In the case of temporal extensions of Datalog, such as Datalog_{1S}, critical datasets need to be further modified to capture the temporal dimension of models (Chomicki & Imielinski, 1988). This is especially challenging in the case of DatalogMTL, where the density of the timeline as well as the form of metric temporal operators occurring in a program must be taken into account.

Our notion of an MTL-acyclic program from Section 6.2 is based on a specific acyclicity notion of metric dependency graphs. Although the concept of acyclicity we proposed, as well as our metric dependency graphs, are specific to DatalogMTL, it is worth noting that numerous types of acyclicity of dependency graphs are commonly used in Datalog research. In particular, the standard notion of acyclicity is used to define non-recursive Datalog programs and more relaxed notions of acyclicity, such as *weak acyclicity*, *super-weak acyclicity*, *model-summarising acyclicity*, and *model-faithful acyclicity* are used to guarantee chase termination in Datalog$^{\exists}$ (Grahne & Onet, 2011; Fagin et al., 2003; Marnette, 2009; Cuenca Grau et al., 2013; Calautti et al., 2015). Furthermore, the acyclicity condition we used to define MTL-acyclic programs is also conceptually related to the techniques used to define temporally acyclic TBoxes in temporalised description logics (Gutiérrez-Basulto et al., 2016a, 2016b), which guarantee that answering atomic queries is feasible in polynomial time.

9. Conclusions and Future Work

In this paper we have proposed and studied finitely materialisable DatalogMTL programs, which are naturally amenable to materialisation-based reasoning via scalable forward chaining techniques. We have studied both data-dependent and data-independent notions of finite materialisability, provided tight complexity bounds and practical materialisation-based algorithms for finite materialisability checking, and proposed two efficiently verifiable sufficient conditions for finite materialisability in the data-independent setting. Finally, we have also shown that fact entailment over finitely materialisable programs is computationally easier than in the general case; in particular, for MTL-acyclic programs, fact entailment has exactly the same complexity as in plain Datalog.

We see many avenues for future work. From a theoretical standpoint we aim to extend our results to include also unbounded programs and datasets. Since most essential results rely on the assumptions that: a program's depth is bounded (e.g., Theorem 24 and Theorem 26), the interval over which a dataset spans is bounded (e.g., Theorem 19, Lemma 32), or materialisation terminates after at most ω steps (e.g., Lemma 37), which all fail in the unbounded case, they cannot be straightforwardly lifted. From a practical perspective, although we have already extended the MeTeoR reasoner with a prototype implementation of our materialisation-based algorithms for finite materialisability checking, evaluation is currently challenging, as there is a lack of real-world DatalogMTL programs that we could use for testing. We will be working with our industrial partners at the SIRIUS Centre for Scalable Data Access to design DatalogMTL programs well-suited for their applications.

Acknowledgements

We are grateful for a fruitful discussion with Michał Pilipczuk on an NL algorithm for checking existence of non-0 weight cycles in weighted graphs, which we used in the proof of Theorem 45 (see also the Appendix).

Our work has been supported by the EPSRC projects OASIS (EP/S032347/1), AnaLOG (EP/P025943/1), and UK FIRES (EP/S019111/1), the SIRIUS Centre for Scalable Data Access, and Samsung Research UK. For the purpose of Open Access, the authors have

References

Abadi, M., & Manna, Z. (1989). Temporal logic programming. *Journal of Symbolic Computation*, *8*(3), 277–295.

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley.

Aguado, F., Cabalar, P., Diéguez, M., Pérez, G., Schaub, T., Schuhmann, A., & Vidal, C. (2021). Linear-time temporal answer set programming. *Theory and Practice of Logic Programming*, *published online*, 1–55.

Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., & Zakharyaschev, M. (2015). First-order rewritability of temporal ontology-mediated queries. In Yang, Q., & Wooldridge, M. J. (Eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25–31, 2015*, pp. 2706–2712. International Joint Conferences on Artificial Intelligence Organization.

Artale, A., Bresolin, D., Montanari, A., Sciavicco, G., & Ryzhikov, V. (2014). DL-Lite and interval temporal logics: a marriage proposal. In Schaub, T., Friedrich, G., & O'Sullivan, B. (Eds.), *ECAI 2014 – 21st European Conference on Artificial Intelligence, 18–22 August 2014, Prague, Czech Republic – Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, Vol. 263 of *Frontiers in Artificial Intelligence and Applications*, pp. 957–958. IOS Press.

Artale, A., & Franconi, E. (1998). A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research*, *9*, 463–506.

Artale, A., & Franconi, E. (2000). A survey of temporal extensions of description logics. *Annals of Mathematics and Artificial Intelligence*, *30*(1), 171–210.

Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., & Zakharyaschev, M. (2017). Ontology-mediated query answering over temporal data: A survey (invited talk). In Schewe, S., Schneider, T., & Wijsen, J. (Eds.), *24th International Symposium on Temporal Representation and Reasoning, TIME 2017, October 16–18, 2017, Mons, Belgium*, Vol. 90 of *LIPIcs*, pp. 1:1–1:37. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Artale, A., Kontchakov, R., Ryzhikov, V., & Zakharyaschev, M. (2014). A cookbook for temporal conceptual data modelling with description logics. *ACM Transactions on Computational Logic*, *15*(3), 1–50.

Artale, A., Kontchakov, R., Ryzhikov, V., & Zakharyaschev, M. (2015a). Interval temporal description logics. In Calvanese, D., & Konev, B. (Eds.), *Proceedings of the 28th International Workshop on Description Logics, Athens,Greece, June 7–10, 2015*, Vol. 1350 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Artale, A., Kontchakov, R., Ryzhikov, V., & Zakharyaschev, M. (2015b). Tractable interval temporal propositional and description logics. In Bonet, B., & Koenig, S. (Eds.),

Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25–30, 2015, Austin, Texas, USA, pp. 1417–1423. AAAI Press.

Baader, F., Borgwardt, S., Koopmann, P., Ozaki, A., & Thost, V. (2017). Metric temporal description logics with interval-rigid names. In Dixon, C., & Finger, M. (Eds.), *Frontiers of Combining Systems – 11th International Symposium, FroCoS 2017, Brasília, Brazil, September 27–29, 2017, Proceedings*, Vol. 10483 of *Lecture Notes in Computer Science*, pp. 60–76. Springer.

Baget, J., Garreau, F., Mugnier, M., & Rocher, S. (2014). Extending acyclicity notions for existential rules. In Schaub, T., Friedrich, G., & O'Sullivan, B. (Eds.), *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, Vol. 263 of *Frontiers in Artificial Intelligence and Applications*, pp. 39–44. IOS Press.

Baudinet, M. (1989). Temporal logic programming is complete and expressive. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '89, pp. 267–280, New York, NY, USA. Association for Computing Machinery.

Beck, H., Dao-Tran, M., Eiter, T., & Folie, C. (2018). Stream reasoning with LARS. *KI - Künstliche Intelligenz*, *32*(2), 193–195.

Bednarczyk, B., Ferens, R., & Ostropolski-Nalewaja, P. (2020). All-instances oblivious chase termination is undecidable for single-head binary tgds. In Bessiere, C. (Ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 1719–1725. International Joint Conferences on Artificial Intelligence Organization.

Bellomarini, L., Nissl, M., & Sallinger, E. (2021). Query evaluation in DatalogMTL – taming infinite query results. *CoRR*, *abs/2109.10691*.

Bellomarini, L., Sallinger, E., & Gottlob, G. (2018). The Vadalog system: Datalog-based reasoning for knowledge graphs. *Proceedings of the VLDB Endowment*, *11*(9), 975–987.

Brandt, S., Kalayci, E. G., Kontchakov, R., Ryzhikov, V., Xiao, G., & Zakharyaschev, M. (2017). Ontology-based data access with a Horn fragment of metric temporal logic. In Singh, S., & Markovitch, S. (Eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4–9, 2017, San Francisco, California, USA*, pp. 1070–1076. AAAI Press.

Brandt, S., Kalaycı, E. G., Ryzhikov, V., Xiao, G., & Zakharyaschev, M. (2018). Querying log data with metric temporal logic. *Journal of Artificial Intelligence Research*, *62*, 829–877.

Bry, F., Eisinger, N., Eiter, T., Furche, T., Gottlob, G., Ley, C., Linse, B., Pichler, R., & Wei, F. (2007). Foundations of rule-based query answering. In Antoniou, G., Aßmann, U., Baroglio, C., Decker, S., Henze, N., Patranjan, P., & Tolksdorf, R. (Eds.), *Reasoning Web, Third International Summer School 2007, Dresden, Germany, September 3–7, 2007, Tutorial Lectures*, Vol. 4636 of *Lecture Notes in Computer Science*, pp. 1–153. Springer.

Brzoska, C. (1998). Programming in metric temporal logic. *Theorertical Computer Science*, *202*(1-2), 55–125.

Cabalar, P. (2022). Temporal ASP: From logical foundations to practical use with telingo. In Šimkus, M., & Varzinczak, I. (Eds.), *Reasoning Web. Declarative Artificial Intelligence*, pp. 94–114, Cham. Springer International Publishing.

Cabalar, P., Diéguez, M., Schaub, T., & Schuhmann, A. (2020). Towards metric temporal answer set programming. *Theory and Practice of Logic Programming*, *20*(5), 783–798.

Cabalar, P., Kaminski, R., Morkisch, P., & Schaub, T. (2019). telingo = asp + time. In Balduccini, M., Lierler, Y., & Woltran, S. (Eds.), *Logic Programming and Nonmonotonic Reasoning*, pp. 256–269, Cham. Springer International Publishing.

Cabalar, P., & Pérez Vega, G. (2007). Temporal equilibrium logic: A first approach. In Moreno Díaz, R., Pichler, F., & Quesada Arencibia, A. (Eds.), *Computer Aided Systems Theory – EUROCAST 2007*, pp. 241–248, Berlin, Heidelberg. Springer.

Calautti, M., Gottlob, G., & Pieris, A. (2015). Chase termination for guarded existential rules. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '15, pp. 91–103, New York, NY, USA. Association for Computing Machinery.

Calautti, M., & Pieris, A. (2021). Semi-oblivious chase termination: The sticky case. *Theory of Computing Systems*, *65*(1), 84–121. Database Theory.

Carral, D., Dragoste, I., González, L., Jacobs, C. J. H., Krötzsch, M., & Urbani, J. (2019). Vlog: A rule engine for knowledge graphs. In Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I. F., Hogan, A., Song, J., Lefrançois, M., & Gandon, F. (Eds.), *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II*, Vol. 11779 of *Lecture Notes in Computer Science*, pp. 19–35. Springer.

Chomicki, J. (1990). Polynomial time query processing in temporal deductive databases. In Rosenkrantz, D. J., & Sagiv, Y. (Eds.), *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, April 2–4, 1990, Nashville, Tennessee, USA*, pp. 379–391. Association for Computing Machinery.

Chomicki, J. (1995). Depth-bounded bottom-up evaluation of logic programs. *Journal of Logic Programming*, *25*(1), 1–31.

Chomicki, J., & Imielinski, T. (1988). Temporal deductive databases and infinite objects. In Edmondson-Yurkanan, C., & Yannakakis, M. (Eds.), *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 21–23, 1988, Austin, Texas, USA*, pp. 61–73. Association for Computing Machinery.

Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., & Wang, Z. (2013). Acyclicity notions for existential rules and their application to query answering in ontologies. *Journal of Artificial Intelligence Research*, *47*, 741–808.

Dantsin, E., Eiter, T., Gottlob, G., & Voronkov, A. (2001). Complexity and expressive power of logic programming. *ACM Computing Surveys*, *33*(3), 374–425.

Deutsch, A., Nash, A., & Remmel, J. B. (2008). The chase revisited. In Lenzerini, M., & Lembo, D. (Eds.), *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9–11, 2008, Vancouver, BC, Canada*, pp. 149–158. Association for Computing Machinery.

Diéguez, M. (2012). Temporal answer set programming. In Dovier, A., & Costa, V. S. (Eds.), *Technical Communications of the 28th International Conference on Logic Programming, ICLP 2012, September 4–8, 2012, Budapest, Hungary*, Vol. 17 of *LIPIcs*, pp. 445–450. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Eiter, T., & Simkus, M. (2009). Bidirectional answer set programs with function symbols. In Boutilier, C. (Ed.), *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11–17, 2009*, pp. 765–771. International Joint Conferences on Artificial Intelligence Organization.

Eiter, T., & Simkus, M. (2010). FDNC: decidable nonmonotonic disjunctive logic programs with function symbols. *ACM Transactions on Computational Logic*, *11*(2), 14:1–14:50.

Fagin, R., Kolaitis, P. G., Miller, R. J., & Popa, L. (2003). Data exchange: Semantics and query answering. In Calvanese, D., Lenzerini, M., & Motwani, R. (Eds.), *Database Theory – ICDT 2003, 9th International Conference, Siena, Italy, January 8–10, 2003, Proceedings*, Vol. 2572 of *Lecture Notes in Computer Science*, pp. 207–224. Springer.

Fagin, R., Kolaitis, P. G., Miller, R. J., & Popa, L. (2005). Data exchange: Semantics and query answering. *Theoretical Computer Science*, *336*(1), 89–124.

Feikin, D. R., Higdon, M. M., Abu-Raddad, L. J., Andrews, N., Araos, R., Goldberg, Y., Groome, M. J., Huppert, A., O'Brien, K. L., Smith, P. G., Wilder-Smith, A., Zeger, S., Deloria Knoll, M., & Patel, M. K. (2022). Duration of effectiveness of vaccines against sars-cov-2 infection and covid-19 disease: Results of a systematic review and meta-regression. *The Lancet*, *399*(10328), 924–944.

Gogacz, T., & Marcinkowski, J. (2014). All-instances termination of chase is undecidable. In Esparza, J., Fraigniaud, P., Husfeldt, T., & Koutsoupias, E. (Eds.), *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8–11, 2014, Proceedings, Part II*, Vol. 8573 of *Lecture Notes in Computer Science*, pp. 293–304. Springer.

Gogacz, T., Marcinkowski, J., & Pieris, A. (2020). All-instances restricted chase termination. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'20, pp. 245–258, New York, NY, USA. Association for Computing Machinery.

Grahne, G., & Onet, A. (2011). On conditional chase termination. In Barceló, P., & Tannen, V. (Eds.), *Proceedings of the 5th Alberto Mendelzon International Workshop on Foundations of Data Management, Santiago, Chile, May 9–12, 2011*, Vol. 749 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Grahne, G., & Onet, A. (2018). Anatomy of the chase. *Fundamenta Informaticae*, *157*(3), 221–270.

Gutiérrez-Basulto, V., Jung, J. C., & Kontchakov, R. (2016a). On decidability and tractability of querying in temporal EL. In Lenzerini, M., & Peñaloza, R. (Eds.), *Proceedings*

of the 29th International Workshop on Description Logics, Cape Town, South Africa, April 22–25, 2016, Vol. 1577 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Gutiérrez-Basulto, V., Jung, J. C., & Kontchakov, R. (2016b). Temporalized EL ontologies for accessing temporal data: Complexity of atomic queries. In Kambhampati, S. (Ed.), *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016*, pp. 1102–1108. International Joint Conferences on Artificial Intelligence Organization.

Gutiérrez-Basulto, V., Jung, J. C., & Ozaki, A. (2016c). On metric temporal description logics. In Kaminka, G. A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., & van Harmelen, F. (Eds.), *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, Vol. 285 of *Frontiers in Artificial Intelligence and Applications*, pp. 837–845. IOS Press.

Immerman, N., & Landau, S. (1989). The complexity of iterated multiplication. In *Proceedings: Fourth Annual Structure in Complexity Theory Conference, University of Oregon, Eugene, Oregon, USA, June 19–22, 1989*, pp. 104–111. IEEE Computer Society.

Kalayci, E. G., Brandt, S., Calvanese, D., Ryzhikov, V., Xiao, G., & Zakharyaschev, M. (2019). Ontology–based access to temporal data with Ontop: A framework proposal. *International Journal of Applied Mathematics and Computer Science*, *29*(1), 17–30.

Karimi, A., Zhang, H., & You, J.-H. (2021). Restricted chase termination for existential rules: A hierarchical approach and experimentation. *Theory and Practice of Logic Programming*, *21*(1), 4–50.

Kikot, S., Ryzhikov, V., Wałęga, P. A., & Zakharyaschev, M. (2018). On the data complexity of ontology-mediated queries with MTL operators over timed words. In Ortiz, M., & Schneider, T. (Eds.), *Proceedings of the 31st International Workshop on Description Logics co-located with 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018), Tempe, Arizona, US, October 27–29, 2018*, Vol. 2211 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Kontchakov, R., Pandolfo, L., Pulina, L., Ryzhikov, V., & Zakharyaschev, M. (2016). Temporal and spatial OBDA with many-dimensional Halpern-Shoham logic. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016*, pp. 1160–1166. International Joint Conferences on Artificial Intelligence Organization.

Kovtunova, A. (2017). *Ontology-mediated query answering with lightweight temporal description logics*. Ph.D. thesis, Free University of Bozen-Bolzano.

Krötzsch, M., Marx, M., & Rudolph, S. (2019). The power of the terminating chase (invited talk). In Barceló, P., & Calautti, M. (Eds.), *22nd International Conference on Database Theory, ICDT 2019, March 26–28, 2019, Lisbon, Portugal*, Vol. 127 of *LIPIcs*, pp. 3:1–3:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Marnette, B. (2009). Generalized schema-mappings: From termination to tractability. In *Proceedings of the Twenty-Eighth ACM SIGMOD-SIGACT-SIGART Symposium on*

Principles of Database Systems, PODS '09, pp. 13–22, New York, NY, USA. Association for Computing Machinery.

Marnette, B., & Geerts, F. (2010). Static analysis of schema-mappings ensuring oblivious termination. In Segoufin, L. (Ed.), *Database Theory – ICDT 2010, 13th International Conference, Lausanne, Switzerland, March 23–25, 2010, Proceedings*, ACM International Conference Proceeding Series, pp. 183–195. Association for Computing Machinery.

Mori, M., Papotti, P., Bellomarini, L., & Giudice, O. (2022). Neural machine translation for fact-checking temporal claims. In Aly, R., Christodoulopoulos, C., Cocarascu, O., Guo, Z., Mittal, A., Schlichtkrull, M., Thorne, J., & Vlachos, A. (Eds.), *Proceedings of the Fifth Fact Extraction and VERification Workshop (FEVER)*, pp. 78–82, Dublin, Ireland. Association for Computational Linguistics.

Motik, B., Nenov, Y., Piro, R., & Horrocks, I. (2019). Maintenance of Datalog materialisations revisited. *Artificial Intelligence*, *269*, 76–136.

Motik, B., Nenov, Y., Piro, R., Horrocks, I., & Olteanu, D. (2014). Parallel materialisation of Datalog programs in centralised, main-memory RDF systems. In Brodley, C. E., & Stone, P. (Eds.), *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27–31, 2014, Québec City, Québec, Canada*, pp. 129–137. AAAI Press.

Nissl, M., & Sallinger, E. (2022). Modelling smart contracts with DatalogMTL. In Ramanath, M., & Palpanas, T. (Eds.), *Proceedings of the Workshops of the EDBT/ICDT*, Vol. 3135 of *CEUR*. CEUR-WS.org.

Ozaki, A., Krötzsch, M., & Rudolph, S. (2018). Happy ever after: Temporally attributed description logics. In Ortiz, M., & Schneider, T. (Eds.), *Proceedings of the 31st International Workshop on Description Logics co-located with 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018), Tempe, Arizona, US, October 27–29, 2018*, Vol. 2211 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Raheb, K. E., Mailis, T., Ryzhikov, V., Papapetrou, N., & Ioannidis, Y. E. (2017). Balonse: Temporal aspects of dance movement and its ontological representation. In Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., & Hartig, O. (Eds.), *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28–June 1, 2017, Proceedings, Part II*, Vol. 10250 of *Lecture Notes in Computer Science*, pp. 49–64.

Reif, J. H., & Tate, S. R. (1992). On threshold circuits and polynomial computation. *SIAM Journal on Computing*, *21*(5), 896–908.

Ronca, A., Kaminski, M., Cuenca Grau, B., & Horrocks, I. (2022). The delay and window size problems in rule-based stream reasoning. *Artificial Intelligence*, *306*, 103668.

Rosser, J. B., & Schoenfeld, L. (1962). Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, *6*(1), 64–94.

Ryzhikov, V., Wałęga, P. A., & Zakharyaschev, M. (2019). Data complexity and rewritability of ontology-mediated queries in metric temporal logic under the event-based semantics. In Kraus, S. (Ed.), *Proceedings of the Twenty-Eighth International Joint Conference*

on Artificial Intelligence, IJCAI 2019, Macao, China, August 10–16, 2019, pp. 1851–1857. International Joint Conferences on Artificial Intelligence Organization.

Ryzhikov, V., Wałęga, P. A., & Zakharyaschev, M. (2020). Temporal ontology-mediated queries and first-order rewritability: A short course. In Manna, M., & Pieris, A. (Eds.), *Reasoning Web. Declarative Artificial Intelligence - 16th International Summer School 2020, Oslo, Norway, June 24–26, 2020, Tutorial Lectures*, Vol. 12258 of *Lecture Notes in Computer Science*, pp. 109–148. Springer.

Schneider, P., Alvarez-Coello, D., Le-Tuan, A., Nguyen-Duc, M., & Le-Phuoc, D. (2022). Stream reasoning playground. In *The Semantic Web: 19th International Conference, ESWC 2022, Hersonissos, Crete, Greece, May 29–June 2, 2022, Proceedings*, pp. 406–424, Berlin, Heidelberg. Springer-Verlag.

Tena Cucala, D. J., Wałęga, P. A., Cuenca Grau, B., & Kostylev, E. V. (2021). Stratified negation in Datalog with metric temporal operators. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2–9, 2021*, pp. 6488–6495. AAAI Press.

Thost, V. (2018). Metric temporal extensions of DL-Lite and interval-rigid names. In Thielscher, M., Toni, F., & Wolter, F. (Eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, October 30–November 2, 2018*, pp. 665–666. AAAI Press.

Wałęga, P. A., Cuenca Grau, B., Kaminski, M., & Kostylev, E. V. (2019). DatalogMTL: Computational complexity and expressive power. In Kraus, S. (Ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10–16, 2019*, pp. 1886–1892. International Joint Conferences on Artificial Intelligence Organization.

Wałęga, P. A., Cuenca Grau, B., Kaminski, M., & Kostylev, E. V. (2020a). DatalogMTL over the integer timeline. In Calvanese, D., Erdem, E., & Thielscher, M. (Eds.), *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12–18, 2020*, pp. 768–777.

Wałęga, P. A., Cuenca Grau, B., Kaminski, M., & Kostylev, E. V. (2020b). Tractable fragments of Datalog with metric temporal operators. In Bessiere, C. (Ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pp. 1919–1925. International Joint Conferences on Artificial Intelligence Organization.

Wałęga, P. A., Kaminski, M., & Cuenca Grau, B. (2019). Reasoning over streaming data in metric temporal Datalog. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27–February 1, 2019*, pp. 3092–3099. AAAI Press.

Wałęga, P. A., Tena Cucala, D. J., Kostylev, E. V., & Cuenca Grau, B. (2021a). DatalogMTL with negation under stable models semantics. In Bienvenu, M., Lakemeyer, G., &

Erdem, E. (Eds.), *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, pp. 609–618.

Wałęga, P. A., Zawidzki, M., & Cuenca Grau, B. (2021b). Finitely materialisable Datalog programs with metric temporal operators. In Bienvenu, M., Lakemeyer, G., & Erdem, E. (Eds.), *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3–12, 2021*, pp. 619–628.

Wang, D., Hu, P., Wałęga, P. A., & Cuenca Grau, B. (2022). MeTeoR: Practical reasoning in Datalog with metric temporal operators. *CoRR*, *abs/2201.04596*.

Appendix A. Proof Details

Lemma. *Checking if there is a simple cycle of non-*0 *weight in a directed graph whose edges are weighted with rational numbers encoded in binary, or with* $-\infty$ *or* ∞ *is feasible in* NL *(assuming that* $\infty - \infty = \infty$*).*

Proof. Let G be the input graph and let n be the size of its representation (we assume that $n > 1$, as otherwise G has no edges). We observe that G has a non-0 weight simple cycle if and only if G has a non-0 weight cycle of length at most n. We will show how to check in NL the latter statement, which does not require verifying that the cycle is simple. If a cycle of G has at least one edge of weight ∞ or $-\infty$, then this cycle has a non-0 weight; clearly, such a cycle can be guessed edge by edge in NL. Thus, we can focus on detecting a cycle whose all edges are weighted with rational numbers. We will guess such a cycle edge by edge, but now we need a more sophisticated approach for verifying that this cycle has a non-0 weight. Note that, while guessing the cycle, we cannot keep its partial weight in memory, since it is not feasible in logarithmic space. Instead, our procedure keeps in memory the (representation of) a partial weight modulo a prime number p that is guessed in NL (so p can be kept in logarithmic memory). More precisely, for a partial weight $\frac{x}{y}$, our procedure keeps in memory its representation in the field $\mathbb{F}_p$ of order p. With a slight abuse of notation we denote such a representation by $x \cdot y^{-1}$ (where $\cdot$ and $^{-1}$ are the multiplication and multiplicative inverse operators in $\mathbb{F}_p$, respectively, and x and y are taken modulo p). Then, we claim that it suffices to verify that the final value in $\mathbb{F}_p$ that we keep in memory is non-0. In particular, we claim that checking existence of a non-0 (rational) weight simple cycle in G reduces to checking existence of a prime number $p \in \{1, \dots, 4n^4\}$ and a cycle in G of length at most n satisfying the following conditions:

(i) none of the denominators of weights in G is divisible by p and

(ii) the representation in $\mathbb{F}_p$ of the weight of the guessed cycle is not 0.

Observe that Condition (i) guarantees that, for each edge weight $\frac{x}{y}$ in G, the representation of y in $\mathbb{F}_p$ does not equal 0, and therefore, y^{-1} is well defined in $\mathbb{F}_p$. Moreover, as adding the representations of two fractions in $\mathbb{F}_p$ yields the representation of their sum in $\mathbb{F}_p$, the representation mentioned in Condition (ii) is well defined in $\mathbb{F}_p$. Showing the above reduction will end the proof since the problem to which we reduce can be solved in NL; indeed, p can

be guessed in NL, and so can the cycle (edge by edge); moreover, checking Condition (i) is feasible in logarithmic space, and so is checking Condition (ii), as operations in $\mathbb{F}_p$ can be performed in logarithmic space.

To show that the reduction is correct, assume first that there exist a prime number p and a cycle σ in G for which Conditions (i) and (ii) hold. Let $\frac{a}{b}$ be the weight of σ, so we need to show that $a \neq 0$. Recall that, by Condition (i), $a \cdot b^{-1}$ is well defined in $\mathbb{F}_p$; we obtain also that $b^{-1} \neq 0$. By Condition (ii), $a \cdot b^{-1} \neq 0$ in $\mathbb{F}_p$, and so, a is not 0 in $\mathbb{F}_p$. Therefore, a is non-0, and so is the weight $\frac{a}{b}$.

For the opposite direction assume that G has a simple cycle σ of some non-0 weight $\frac{a}{b}$ and length at most n. We will show existence of a prime number $p \in \{1, \ldots, 4n^4\}$ such that neither a nor any of the denominators of weights in G is divisible by p. Each of these denominators uses at most n bits in binary representation, so each denominator has a value bounded by 2^n and at most n prime divisors (as each such prime divisor is at least 2). As there are at most n denominators, the total number of their prime divisors is at most n^2. Since a is the sum of at most n components whose values are bounded by 2^n, the value of a is bounded by $n \cdot 2^n$, and so, a has at most $\log_2(n) + n$ prime divisors. Thus, the number of primes that divide a or any of the denominators in G is bounded by $2n^2$. It is known that for any $x \geq 17$, we have $\pi(x) > \frac{x}{\log_2(x)}$, where $\pi(x)$ is the number of primes not larger than x (Rosser & Schoenfeld, 1962, Corollary 1). We observe that $4n^4 > 17$ (since $n > 1$), so $\pi(4n^4) > \frac{4n^4}{\log_2(4n^4)}$. As $\log_2(4n^4) < \sqrt{4n^4}$ (since $4n^4 > 16$), we have $\frac{4n^4}{\log_2(4n^4)} > \frac{4n^4}{\sqrt{4n^4}} = 2n^2$. We obtain that $\pi(4n^4) > 2n^2$, so there must exist a prime number $p \in \{1, \ldots, 4n^4\}$ that divides neither a nor any denominator in G. Note that we have shown that our p satisfies Condition (i), so it remains to show that p and σ satisfy also Condition (ii). Since neither a nor any of the denominators of weights in G is divisible by p, we have $a \neq 0$ and $b^{-1} \neq 0$ in $\mathbb{F}_p$. Consequently, $a \cdot b^{-1} \neq 0$ in $\mathbb{F}_p$, so Condition (ii) holds. □