Reasoning About Time in DatalogMTL: Course Notes

Przemysław Andrzej Wałęga
Queen Mary University of London, UK
University of Łódź, Poland

Abstract

Many real-world applications, such as those in healthcare, finance, and logistics, require reasoning over temporal data. Standard rule-based languages like Datalog, however, lack explicit mechanisms for handling time and temporal dependencies. In this chapter, we discussDatalogMTL, an extension of Datalog with operators frommetric temporal logic that allow to express complex temporal properties. We focus on reasoning algorithms for DatalogMTL, discussing bothmaterialisation, based on fixpoint applications of the immediate consequence operator, and anovel saturation-based extensionthat detects and halts infinite derivations, ensuring both completeness and termination of reasoning.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Programming logic

Keywords and phrases DatalogMTL, Logic Programming, Temporal Reasoning

Digital Object Identifier 10.4230/OASIcs.RW.2024/2025.9

Category Invited Paper

Funding This research is partially funded by the European Union (ERC, ExtenDD, project number: 101054714). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Acknowledgements These course notes are based on the author's research on DatalogMTL, previously presented in several co-authored papers [54, 53, 55]. The author is deeply grateful to all colleagues with whom he had the pleasure of collaborating on this topic, in particular Bernardo Cuenca Grau and Michał Zawidzki, whose contributions were essential for establishing the results discussed here.

1 All We Need Is Time

Many application domains, such as healthcare, finance, and logistics are inherently temporal. Events like medical appointments or financial transactions take place in time, and reasoning about them requires explicit consideration of the temporal dimension. Datalog – a state-of-the-art rule-based language for reasoning over structured data [1, 33, 22] – is, however, atemporal. It allows us to write rules such as

$$\mathit{NegTest}(x) \leftarrow \mathit{Immune}(x)$$

stating that if a patient x is immune, their test needs to be negative. However, such rules do not allow us to represent the temporal relation between events, for example that a patient needs to be immune continuously in the last five days to be guaranteed to have a negative test. How can we perform temporal reasoning in Datalog?

One approach is to enrich Datalog with numbers representing points of time, and with arithmetic operators allowing to express dependencies between time points. For example one could write the following rule

$$\mathit{NegTest}(x, t_1 + 5, t_2) \leftarrow \mathit{Immune}(x, t_1, t_2) \wedge (t_1 \leq t_2 + 5)$$

stating that if a patient is immune continuously within a time interval $[t_1, t_2]$ of length 5 at least, then they will have a negative result of a test performed within the time interval $[t_1 + 5, t_2]$. Available Datalog engines such as Nemo [27], RDFox [33], and Vadalog [9] often allow for exploiting arithmetic operators, but such an extension quickly leads to undecidability of reasoning.

Another approach, which will be the focus of this course, is to design extensions of Datalog tailored specifically for temporal reasoning. Instead of arithmetic operators, they use temporal operators. For example, we can use an operator $\boxminus_{[0,5]}$, stating that an event happened continuously for the past interval of length 5, to write the following rule:

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

Such temporal languages are carefully crafted to balance expressive power with computational complexity. In particular, they aim to capture rich temporal properties while ensuring that reasoning remains decidable and can be carried out algorithmically.

Among temporal extensions of Datalog, we will focus on DatalogMTL [12, 46], which augments Datalog with operators from Metric Temporal Logic (MTL) [29] such as already presented $\boxminus_{[0,5]}$. Such operators have high expressive power, and so, DatalogMTL subsumes many other temporal dialects of Datalog. DatalogMTL provides a versatile temporal reasoning language with applications in areas such as Semantic Web, Stream Reasoning, and Knowledge Graphs. Among others, DatalogMTL has been applied in ontology-based data access [11], economic fact-checking [31], human movement description [35], and verification of banking agreements [34].

To provide a better intuition about DatalogMTL, let us illustrate the language with some examples of facts and rules. DatalogMTL facts are annotated with time points (or time intervals) in which they hold, and rules allow us to express temporal dependencies between events.

▶ **Example 1.** Consider an exemplary DatalogMTL dataset, denoted by $\mathcal{D}_{\mathsf{ex}}$, which we will use throughout this paper. It contains two facts:

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

stating that Ben got vaccinated at the time point $199\frac{2}{3}$ and that he had no symptoms continuously within the interval $(181, 320\frac{1}{2}]$.

Note that the timeline in DatalogMTL is an abstract sequence of rational time points. Nevertheless, it is easy to give it an intuitive interpretation. For instance if we assume that the first day (of the current year) in our database corresponds to the interval $(0, 1]$, the second day to $(1, 2]$, and so on, then facts in $\mathcal{D}_{\mathsf{ex}}$ state that Ben was vaccinated at 4 p.m. on July 19 and that he had no symptoms since midnight on July 1 until noon on August 30.

Rules in DatalogMTL, in turn, extend classical Datalog with temporal operators, which enable reasoning over the temporal dimension of data. These operators are indexed by time intervals capturing the metric dependencies. In what follows we will provide examples of temporal operators that can be used in DatalogMTL. First, we present the "diamond" operators $\Diamond$ – for "sometime in the future" and $\Diamond$ – for "sometime in the past". With such operators we can write the following expressions:

- $\Diamond_{[1,2]}\, \mathit{Vaccinated}$, which states that *Vaccinated* will hold at some point between 1 and 2 time units in the future. More precisely, this expression is true at a time point t if there exists a moment within the interval $[t + 1, t + 2]$ at which *Vaccinated* holds.
- $\Diamond_{[1,2]}\, \mathit{Vaccinated}$, which uses the past version $\Diamond$ of $\Diamond$. The formula $\Diamond_{[1,2]}\, \mathit{Vaccinated}$ holds at all time points t such that *Vaccinated* holds at some time point in the interval $[t - 2, t - 1]$.

DatalogMTL allows also to use "box" versions of the above "diamond" operators, for instance:

- $\boxplus_{[1,2]}\mathit{Vaccinated}$ holds true at time points t such that $\mathit{Vaccinated}$ holds continuously in the interval $[t+1, t+2]$.
- $\boxminus_{[1,2]}\mathit{Vaccinated}$ holds true at time points t such that $\mathit{Vaccinated}$ holds continuously in the interval $[t-2, t-1]$.

Finally, there are also metric version of "until" and "since" operators as presented below:

- $\mathit{NoSympt}\,\mathcal{U}_{[1,2]}\,\mathit{Vaccinated}$ holds true at time points t such that $\mathit{Vaccinated}$ holds at some time point $t' \in [t+1, t+2]$ and $\mathit{NoSympt}$ holds from t until this t'.
- $\mathit{NoSympt}\,\mathcal{S}_{[1,2]}\,\mathit{Vaccinated}$ holds true at time points t such that $\mathit{Vaccinated}$ holds at some time point $t' \in [t-2, t-1]$ and $\mathit{NoSympt}$ holds from t since this t'.

We will provide formal semantics of all these operators in the following section, but it is already worth presenting what kind of reasoning scenarios DatalogMTL allows us to express. To this end, consider the following example that models how immunity may arise either from vaccination or from prior exposure to a disease.

▶ **Example 2.** Consider a scenario where evidence suggests that immunity to a disease lasts for at least 90 days under the following conditions: an individual has been vaccinated and, within 3–4 weeks afterwards, either remained asymptomatic or tested negative. Also, an individual is immune if they were infected within the past six months (183 days), excluding the most recent ten days during which they showed no symptoms. In addition, individuals who have been immune for at least five days are observed to test negative. These conditions can be formalised in a DatalogMTL program Π_{ex} consisting of the following 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, 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 $\diamondminus_{[21,28]}$). 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 $\boxminus_{[0,10]}$). Finally, Rule (4) states that if an individual was immune continuously for the last 5 days (operator $\boxminus_{[0,5]}$), they will have a negative test.

As the example above shows, DatalogMTL allows us to express quite complex temporal rules. The main question we will consider is how to construct reasoning algorithms for this language?

We will begin in Section 2 with a formal presentation of the syntax and semantics of DatalogMTL. In Section 3, we study materialisation as a reasoning mechanism for DatalogMTL and identify the cases where it yields a complete reasoning procedure. However, materialisation alone often proves insufficient. This motivates Section 4, where we present the technically most challenging result: a saturation mechanism that detects loops in infinite derivations and thereby enables complete reasoning even when pure materialisation fails. Finally, in Section 5, we discuss related work on DatalogMTL, provide references for further reading, and highlight several interesting open problems in this research area.

2 Getting Formal with DatalogMTL

In this section, we will provide a formal definition of syntax and semantics of DatalogMTL, as well as introduce the standard reasoning tasks in this setting. For crucial notions, we will provide examples and illustrations to help the reader become familiar with them, as their understanding is essential for the subsequent sections.

2.1 Timeline and Time Intervals

We focus on the original setting for DatalogMTL, where the timeline is dense and consists of rational numbers. However, it is worth noting that DatalogMTL was also considered over the discrete integer timeline [47]. Formally, we let the *(rational) timeline* be the set $\mathbb{Q}$ of rational (both positive and negative) numbers and we will call each element of the timeline a *time point*. An *interval* ϱ is a non-empty subset of $\mathbb{Q}$ satisfying the following standard properties:

- 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
- 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 (i.e., $\varrho \subseteq \mathbb{Q}_{\geq 0}$), and it is *bounded* if both its left and right endpoints are rational numbers.

We use the standard representation $\langle \varrho^-, \varrho^+ \rangle$ for intervals 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. For example, the following represent three different intervals: $[2\frac{1}{2}, 4]$, $(2\frac{1}{2}, 4]$, and $[2\frac{1}{2}, 4)$. 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. For example instead of $[2\frac{1}{2}, 2\frac{1}{2}]$ we may simply write $2\frac{1}{2}$.

2.2 Syntax of DatalogMTL

We let a *relational atom* be 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). For example $\mathit{Vaccinated}(\mathsf{Ben})$ is a relational atom with a predicate *Vaccinated* of arity 1 and with a single constant Ben, whereas $\mathit{Vaccinated}(x)$ is a relational atom using a variable x instead of a constant. We allow also for predicates of arity 0, such as *Sunny*, which do not require terms. A *metric atom* is an extension of a relational atom with metric temporal operators. In particular, 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:

$$M ::= \top \mid \bot \mid P(\mathbf{s}) \mid \Diamond_\varrho M \mid \Diamondblock_\varrho M \mid \Box_\varrho M \mid \boxplus_\varrho M \mid M \mathcal{S}_\varrho M \mid M \mathcal{U}_\varrho M.$$

For example $\Diamondblock_{(10,183]} \mathit{Infected}(x)$ is a metric atom. Notice that the definition allows also for arbitrary nesting of temporal operators, so $\boxplus_{[11,14]} \Diamondblock_{(10,183]} \mathit{Infected}(x)$ is also a metric atom.

A *rule* is an expression built from metric atoms as follows:

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

where each M_i is a metric atom and M' is a metric atom allowing only for "boxes" among

temporal operators, namely M' is generated by the following grammar:[1]

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

The fact that we disallow some temporal operators in M' is dictated by complexity results; if we allow for M' being an arbitrary metric atom, reasoning would become undecidable. 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*. As an example of DatalogMTL rules, consider Rules (1)–(4) from Π_{ex} in Example 2. A *program*, like Π_{ex}, is a finite set of rules[2].

An expression (e.g. a metric atom, rule, or program) is *ground* if it mentions no variables. Hence *Vaccinated*(Ben) is ground, but *Vaccinated*(x) is not. A *fact* over an interval ϱ is an expression $M@\varrho$, with M a ground relational atom. A *dataset*, for example $\mathcal{D}_{\mathsf{ex}}$ from Example 1, is a finite set of facts.

The *grounding* of a program Π, written as $\mathsf{ground}(\Pi)$, is the ground program obtained as the set of all ground rules that can be obtained by assigning constants 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 $\mathsf{ground}(\Pi)$ rules which do mention only those constants which are present in Π or $\mathcal{D}$.

2.3 Semantics of DatalogMTL

We will focus in this course on the standard *continuous semantics* of DatalogMTL, as opposed to the alternative pointwise semantics [36]. In the standard setting 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 complex ground metric atoms as given in Table 1.

Table 1 Semantics of ground metric atoms.

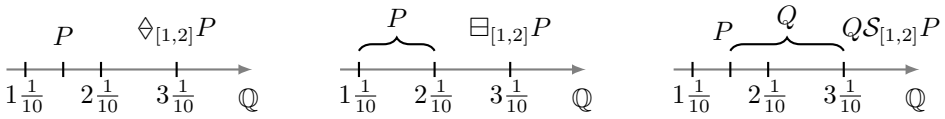
$\mathfrak{I}, t \models \top$		for each t
$\mathfrak{I}, t \models \bot$		for no t
$\mathfrak{I}, t \models \diamondminus_\varrho M$	iff	$\mathfrak{I}, t' \models M$ for some t' with $t - t' \in \varrho$
$\mathfrak{I}, t \models \diamondplus_\varrho M$	iff	$\mathfrak{I}, t' \models M$ for some t' with $t' - t \in \varrho$
$\mathfrak{I}, t \models \boxminus_\varrho M$	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 $t'' \in (t, t')$

The meaning of past operators $\diamondminus$, $\boxminus$, and $\mathcal{S}$ is additionally visualised in figures below. The first figure shows that $\diamondminus_{[1,2]} P$ holds at $3\frac{1}{10}$ if P holds at some time point located within the interval $[1\frac{1}{10}, 2\frac{1}{10}]$. The second figure indicates that $\boxminus_{[1,2]} P$ holds at $3\frac{1}{10}$ if P holds continuously in the interval $[1\frac{1}{10}, 2\frac{1}{10}]$. The third figure, in turn, shows that $Q\mathcal{S}_{[1,2]} P$ holds at

[1] Notice that we disallow $\bot$ in M', which ensures that no inconsistency can occur; DatalogMTL is often considered with rules allowing for $\bot$ in M', but to simplify the setting we will not consider such rules.

[2] Precisely speaking we require also that rules are *safe*, namely each variable mentioned in the head of a rule occurs also in its body, and this occurrence is not in a left operand of $\mathcal{S}$ or $\mathcal{U}$; without this restriction we could write rules like $Q(x) \leftarrow P$, which are usually disallowed in Datalog.

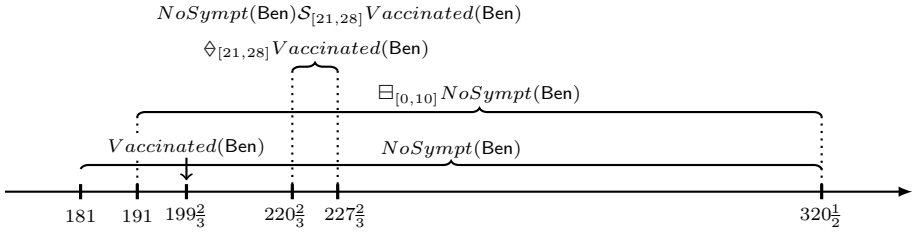
$3\frac{1}{10}$ if P holds at some time point within $[1\frac{1}{10}, 2\frac{1}{10}]$ and Q holds continuously from this time point up to the time point $3\frac{1}{10}$. The meaning of future operators $\Diamond\!\!\!\!+$, $\boxplus$, and $\mathcal{U}$ is symmetric to the meaning of the corresponding past operators $\Diamond$, $\boxminus$, and $\mathcal{S}$, respectively.



To illustrate the notion of satisfaction in an interpretation, consider the following example.

▶ **Example 3.** Consider an interpretation $\mathfrak{I}$ represented by the dataset $\mathcal{D}_{\mathsf{ex}}$ from Example 1, that is, $\mathfrak{I}, 199\frac{2}{3} \models \mathit{Vaccinated}(\mathsf{Ben})$ and $\mathfrak{I}, t \models \mathit{NoSympt}(\mathsf{Ben})$, 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):

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



We say that an interpretation $\mathfrak{I}$ *satisfies a 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 its every ground instance, that is, 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 dataset $\mathcal{D}$ if $\mathfrak{I}$ satisfies each fact in $\mathcal{D}$. A dataset $\mathcal{D}$ *entails* a fact $M@\varrho$ if each model of $\mathcal{D}$ is also a model of $M@\varrho$. A program Π and a dataset $\mathcal{D}$ *entail* a fact $M@\varrho$, written $(\Pi, \mathcal{D}) \models M@\varrho$, if each model of both Π and $\mathcal{D}$ is also a model of $M@\varrho$.

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 $\mathfrak{I} = \mathfrak{I}_\mathcal{D}$.

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 $T^{\alpha}_{\Pi}(\mathfrak{I}_{\mathcal{D}})$, for ordinals α, as follows:

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

The *canonical interpretation* $\mathfrak{C}_{\Pi,\mathcal{D}}$ of Π and $\mathcal{D}$ is the interpretation $T^{\omega_1}_{\Pi}(\mathfrak{I}_{\mathcal{D}})$, where ω_1 is the first uncountable ordinal. It is known that $\mathfrak{C}_{\Pi,\mathcal{D}}$ is the least model of Π and $\mathcal{D}$ [11]. In other words, to compute the least model of Π and $\mathcal{D}$, we can apply T_{Π} until a fixpoint is reached, that is, no more information can be derived.

To illustrate the process of applying T_{Π}, which is the basic reasoning mechanism in DatalogMTL, let us consider the following example.

▶ **Example 4.** Consider the program Π_{ex} and dataset $\mathcal{D}_{\mathsf{ex}}$ from Examples 1 and 2. The interpretations $T^{\alpha}_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$ obtained by applying T_{Π} correspond to 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 $T^{2}_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}}) = T^{3}_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$, the fixpoint is reached, and so further applications of T_{Π} cannot derive any new information. Therefore, 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 $\alpha \geq 2$. Hence $T^{2}_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}}) = \mathfrak{C}_{\Pi_{\mathsf{ex}},\mathcal{D}_{\mathsf{ex}}}$ is the canonical interpretation of Π_{ex} and $\mathcal{D}_{\mathsf{ex}}$.

2.4 Reasoning Problems

The main reasoning problem in DatalogMTL we will consider is *fact entailment*: the problem of checking whether a program and a dataset entail a given relational fact. For instance, Example 4 shows that Π_{ex} and $\mathcal{D}_{\mathsf{ex}}$ logically entail $\mathit{NegTest}(\mathsf{Ben})@[225\frac{2}{3}, 317\frac{2}{3}]$, because this fact holds in the canonical interpretations $\mathfrak{C}_{\Pi_{\mathsf{ex}},\mathcal{D}_{\mathsf{ex}}}$. Indeed, it was derived after two applications of the immediate consequence operator.

Another standard problem is *consistency checking*, which is to check whether a given program and a dataset admit a common model. Notice that our definition of a DatalogMTL rule does not allow for $\bot$ to occur in a head. As a result, in our setting every program Π, together with any dataset $\mathcal{D}$ admit a common model, and the canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$ is the minimal among their models. However, if we allowed for $\bot$ in rule heads, which is often considered in the literature, some pairs of a program and dataset would not admit models. For instance this would be the case for a dataset with a single fact $P@[0, 1]$ and a program with a single rule $\bot \leftarrow P$, which clearly leads to inconsistency (derivation of $\bot$).

In the passing, we observe that if we allow for $\bot$ in rule heads, then consistency checking and fact entailment in DatalogMTL reduce to the complements of each other. Specifically, to check whether a program Π and a dataset $\mathcal{D}$ are inconsistent, it suffices to check whether they entail fact $P@0$, for P a fresh atom occurring neither in Π nor in $\mathcal{D}$. Indeed, $P@0$ is entailed, that is $P@0$ holds in all models of Π and $\mathcal{D}$ only if Π and $\mathcal{D}$ have no models, that is, if Π and $\mathcal{D}$ are not consistent. In turn, to check whether Π and $\mathcal{D}$ entail a fact $P(\mathbf{c})@\varrho$, it suffices to check whether the following program and dataset are inconsistent, with P' a

fresh predicate of the same arity as P, $\mathbf{x}$ a tuple of distinct variables, and t an arbitrary time point belonging to the interval ϱ:

$$\Pi' = \Pi \cup \{\bot \leftarrow P'(\mathbf{x}) \wedge \boxminus_{\varrho_1} P(\mathbf{x}) \wedge \boxplus_{\varrho_2} P(\mathbf{x})\}, \qquad \mathcal{D}' = \mathcal{D} \cup \{P'(\mathbf{c})@t\}.$$

Intervals ϱ_1 and ϱ_2 are constructed using ϱ and t; for example, if $\varrho = [t_1, t_2)$, then $\varrho_1 = [0, t - t_1]$ and $\varrho_2 = [0, t_2 - t)$, whereas if $t_2 = \infty$, then $t_2 - t$ stands for ∞. As an exercise, we invite the reader to verify the correctness of the reduction when $\varrho = [t_1, t_2)$, namely that Π and $\mathcal{D}$ entail $P(\mathbf{c})@\varrho$ if and only if Π' and $\mathcal{D}'$ are inconsistent.

It is also worth to mention known computational complexity results. Fact entailment in DatalogMTL is of high complexity; it is ExpSpace-complete [12] and PSpace-complete when measured with respect to the size of a dataset, that is, in data complexity [46]. Tractability in data complexity can be achieved by disallowing certain metric operators and restricting the form of rules [48].

In the remaining sections we will focus on algorithms for checking entailment. In particular, we will be interested in constructing the canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$ for a given program Π and dataset $\mathcal{D}$. Once this is done, entailment of any fact can be checked by simply verifying if $\mathfrak{C}_{\Pi,\mathcal{D}}$ contains this fact or not. However, as we will show in the next section, construction of the canonical interpretation in DatalogMTL is a very non-trivial task.

3 The Never-Ending Story of Materialisation

In this section, we examine the application of standard naïve materialisation as a reasoning technique for DatalogMTL. While materialisation is highly efficient, unlike in classical Datalog it is not guaranteed to terminate in DatalogMTL. We complement this negative result with a positive one, identifying classes of DatalogMTL programs for which materialisation is guaranteed to terminate and thus yields a complete reasoning algorithm.

3.1 Naïve Materialisation

Materialisation-based reasoning is the most common technique of choice implemented in scalable Datalog reasoners [13, 32, 19, 9]. Facts entailed by a program and a dataset are derived in successive rounds of rule applications, mimicking applications of the immediate consequence operator. The process continues until a fixpoint is reached and a *full materialisation* – representing the canonical interpretation – has been computed. At this point, entailment (or answer to more complicated queries) can be checked directly over the computed materialisation.

In this section, we formulate a (possibly non-terminating) generic reasoning procedure for DatalogMTL based on materialisation. In this process, rules are applied by first identifying maximal intervals in which all ground body atoms of a rule hold simultaneously, and then determining intervals in which the head atoms should be derived. Details of this process are presented in Algorithm 1, which performs a single round of rule applications for a given input program and dataset.

To familiarise the reader with this algorithm, let us consider the following example.

▶ **Example 5.** Given the program Π_{ex} and dataset $\mathcal{D}_{\mathsf{ex}}$, Algorithm 1 applies Rule (1) by first identifying $\varrho = [220\frac{2}{3}, 227\frac{2}{3}]$ as the subset-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.

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 $\quad M_1, \dots, M_n \coloneqq$ the body atoms of r;
4 $\quad M' \coloneqq$ the head of r;
5 $\quad P(\mathbf{c}) \coloneqq$ the relational atom in M';
6 $\quad$ **for** *each* $i \in \{1, \dots, n\}$ **do** $\mathsf{I}_{M_i} \coloneqq \big\{ \subseteq\text{-maximal interval } \varrho \mid \mathcal{D} \models M_i @ \varrho \big\}$;
7 $\quad \mathsf{I}_{M'} \coloneqq \big\{ \subseteq\text{-maximal } \varrho \mid \varrho \subseteq (\varrho_1 \cap \dots \cap \varrho_n), \text{ for some } \varrho_1 \in \mathsf{I}_{M_1}, \dots, \varrho_n \in \mathsf{I}_{M_n} \big\}$;
8 $\quad \mathsf{I}_{P(\mathbf{c})} \coloneqq \big\{ \subseteq\text{-maximal } \varrho \mid M' @ \varrho' \models P(\mathbf{c}) @ \varrho, \text{ for some } \varrho' \in \mathsf{I}_{M'} \big\}$;
9 $\quad \mathcal{N} \coloneqq \mathcal{N} \cup \big\{ P(\mathbf{c}) @ \varrho \mid \varrho \in \mathsf{I}_{P(\mathbf{c})} \big\}$; // update the set $\mathcal{N}$
10 **return** $\mathcal{D} \cup \mathcal{N}$;

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 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 6.** *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 $T_\Pi(\mathfrak{I}_\mathcal{D})$, 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 $\mathcal{D} \cup \big\{P(\mathbf{c})@\varrho \mid \varrho \in \mathsf{I}_{P(\mathbf{c})}\big\}$ 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, $\mathcal{D} \cup \big\{P(\mathbf{c})@\varrho \mid \varrho \in \mathsf{I}_{P(\mathbf{c})}\big\}$ 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 $T^i_\Pi(\mathfrak{I}_\mathcal{D})$.

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** $\mathcal{D}_{\mathsf{new}}$;

The following Proposition 7 is a direct consequence of Proposition 6.

▶ **Proposition 7.** *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 our exemplary program Π_{ex} and dataset $\mathcal{D}_{\mathsf{ex}}$ 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. This is why we refer to it as a Procedure rather than an Algorithm. Indeed, for some programs and datasets, the construction of the canonical interpretation requires infinitely many applications of the immediate consequence operator. To illustrate such a behaviour consider the following example.

▶ **Example 8.** 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 (recall that ω is the first infinite ordinal, corresponding to the order type of the natural numbers $\mathbb{N}$) applications of T_Π to $\mathfrak{I}_\mathcal{D}$.

An interesting question arises: for which DatalogMTL programs and datasets, application of the immediate consequence operator reaches a fixpoint in finitely many steps? In other words, when Procedure 2 is guaranteed to terminate, and so, it becomes an algorithm? Notice that this is an important question, as the answer would allow us to determine in which cases the procedure computes the full materialisation and becomes a decision procedure that can be used for checking entailment.

3.2 Finite Materialisability

To answer the questions from the end of the previous subsection, we will introduce the notion of *finitely materialisable* DatalogMTL *programs*. Intuitively, those are programs for which applications of the immediate consequence operator (or equivalently materialisation steps) allow us to obtain a fixpoint in a finite number of steps.

▶ **Definition 9.** *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 $T^\alpha_\Pi(\mathfrak{I}_\mathcal{D}) = \mathfrak{C}_{\Pi,\mathcal{D}}$.*

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 in finite number of steps on input Π and $\mathcal{D}$).*

It follows from Proposition 7 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 $\mathcal{D}$, as we show next.

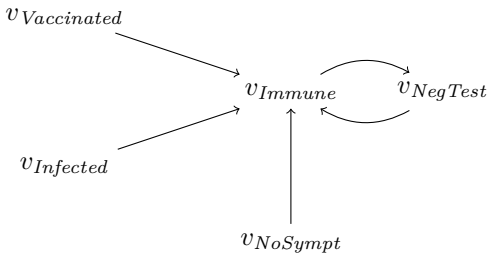
▶ **Example 10.** As shown in Example 4, program Π_{ex} is finitely materialisable for dataset $\mathcal{D}_{\mathsf{ex}}$, and the dataset representing $T^2_{\Pi_{\mathsf{ex}}}(\mathfrak{I}_{\mathcal{D}_{\mathsf{ex}}})$ also represents the canonical interpretation. In contrast, if we consider program $\Pi = \{\boxplus_{[0,1]} P \leftarrow P\}$ and dataset $\mathcal{D} = \{P@[0,1]\}$ from Example 8, 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 9 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.

▶ **Definition 11.** *A program* Π *is called* finitely materialisable *if* Π *is finitely materialisable for all dataset.*

Which programs satisfy this stronger notion of finite materialisability? As we show next, any *non-recursive* DatalogMTL program is guaranteed to be finitely materialisable.

Intuitively non-recursive programs are programs whose rules do not contain derivation loops such as in the rules $Q \leftarrow P$ and $P \leftarrow Q$, where derivation of P allows to derive Q, which allows to derive P again, etc. Formally, (non)recursive programs are defined using the notion of the *dependency graph* of a program. The dependency graph of Π is a directed graph with a vertex v_P for each 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. For example the dependency graph of our Π_{ex} is depicted below.



A program is said to be *recursive*, if its dependency graph is cyclic. Clearly the dependency graph above is cyclic, which implies that Π_{ex} is a recursive program.

Next, we will show that each non-recursive program Π is finitely materialisable, by establishing a bound on the number of rounds of rule applications until the canonical interpretation is reached. In particular, we will show that this bound is given by the number $\mathsf{pred}(\Pi)$ of predicates occurring in Π

▶ **Proposition 12.** *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, $T_\Pi^0(\mathfrak{I}_\mathcal{D}) \models P(\mathbf{c})@t$.

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 $T_\Pi^0(\mathfrak{I}_\mathcal{D}) \models P(\mathbf{c})@t$, and so, $T_\Pi^{d(P)}(\mathfrak{I}_\mathcal{D}) \models P(\mathbf{c})@t$. 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 $T_\Pi^{d(P)-1}(\mathfrak{I}_\mathcal{D})$. Consequently, $T_{\{r\}}\left(T_\Pi^{d(P)-1}(\mathfrak{I}_\mathcal{D})\right) \models P(\mathbf{c})@t$, and so, $T_\Pi^{d(P)}(\mathfrak{I}_\mathcal{D}) \models P(\mathbf{c})@t$. ◀

Moreover, the following example shows that the bound on the number of applications of the immediate consequence operator in Proposition 12 is optimal.

▶ **Example 13.** Consider an arbitrary $n \in \mathbb{N}$ and a program Π_n comprising the rules

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

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 12, cannot be decreased.

Both data-dependent and data-independent notions of finite materialisability in DatalogMTL have been studied in depth, giving rise to a number of interesting results. It is worth mentioning that tight complexity bounds and algorithms for finite materialisability checking have been established. Moreover, efficiently verifiable sufficient conditions for finite materialisability has been provided [53, 54].

To conclude this section, let us recall that materialisation is an iterative procedure that applies the immediate consequence operator repeatedly to the input dataset. Unlike in atemporal Datalog, materialisation in DatalogMTL is not guaranteed to terminate. Nevertheless, one can check whether a given program is finitely materialisable, that is, whether materialisation on this program is guaranteed to halt after finitely many steps. In particular, as we have shown, all non-recursive DatalogMTL programs enjoy this property.

In the next section, we turn to a different reasoning approach for DatalogMTL. Rather than restricting the language to ensure termination of materialisation, we show a more sophisticated mechanism that can detect infinite derivations and exploit this information to compute the full materialisation using a terminating algorithm.

4 Taming the Infinite: The Saturation Algorithm

In this section we present a more sophisticated reasoning algorithm for DatalogMTL. This algorithm [55, 3] will perform standard materialisation steps, but instead of running until a fixpoint is reached, it halts once a partial materialisation satisfies a condition we call *saturation*. As we will show, saturation is guaranteed to occur after a bounded number of materialisation steps, and the full canonical interpretation can then be reconstructed from the saturated partial materialisation. As a result, we will obtain a terminating procedure that is both sound and complete.

A key advantage of this approach is that it does not require syntactic restrictions on the interplay of rules, such as the absence of recursion discussed in the previous section, nor does it rely on the assumption that the program is finitely materialisable. The only requirement is that all intervals in the program and dataset are bounded, that is, they do not use ∞ or $-\infty$ as interval endpoints. This restriction still captures a large and expressive fragment of DatalogMTL. Notice, in particular, that our Π_{ex} from Example 2 is a bounded program. Moreover, it is known that complexity-wise reasoning in bounded DatalogMTL is as hard as reasoning in the unrestricted language [53]. Extending the algorithm to the setting of unbounded intervals remains an interesting open challenge.

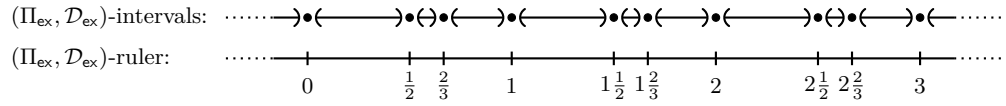
In the following subsections we will observe that canonical interpretations have a regular form. Then we will introduce the concept of saturated interpretations and their unfoldings into canonical interpretations. Finally, we will show a full reasoning algorithm.

4.1 $(\Pi, \mathcal{D})$-ruler and $(\Pi, \mathcal{D})$-intervals

The aim of this short subsection is to show an interesting result about the regular form of the canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$ [46]. The main concepts that we will use to describe this regularity are those of a $(\Pi, \mathcal{D})$-*ruler* and the corresponding $(\Pi, \mathcal{D})$-*intervals*.

▶ **Definition 14.** *For a program Π and a dataset $\mathcal{D}$, 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 $\mathsf{div}(\Pi) = \frac{1}{m}$, where m 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 $m = 1$, and hence, $\mathsf{div}(\Pi) = 1$). A* $(\Pi, \mathcal{D})$-interval*, in turn, 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.*

▶ **Example 15.** Consider the program Π_{ex} and the dataset $\mathcal{D}_{\mathsf{ex}}$ from Example 2. 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.



Now, we say that an interpretation $\mathfrak{I}$ is a $(\Pi, \mathcal{D})$-*interpretation* if, for every 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. In other words, relational atoms hold uniformly over the $(\Pi, \mathcal{D})$-intervals. The main result, we will need in the remaining parts of this section, is that the canonical interpretation and all partial materialisations are $(\Pi, \mathcal{D})$-interpretations [46] as stated next.

▶ **Theorem 16.** *The canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$ of a program Π and a dataset $\mathcal{D}$, as well as the interpretations $T^{\alpha}_{\Pi}(\mathfrak{I}_{\mathcal{D}})$ for any ordinal α, are $(\Pi, \mathcal{D})$-interpretations.*

The proof of this statement is not hard – it can be obtained by induction on α, by showing that $\mathfrak{I}_{\mathcal{D}}$ is a $(\Pi, \mathcal{D})$-interpretation, and each application of the immediate consequence operator leads to a $(\Pi, \mathcal{D})$-interpretation. We invite the reader to write a full proof as an exercise.

In the next subsection we will use the above observation about the canonical intepretation structure to define saturated interpretations.

4.2 Saturated Interpretations and Their Unfoldings

The main aim of this subsection is to present saturation conditions stating the properties that a partial materialisation $T^{k}_{\Pi}(\mathfrak{I}_{\mathcal{D}})$ needs to satisfy so that the it can be used to recover the full canonical interpretation. We call an interpretation satisfying these conditions *saturated* and show how to compute relevant periods and exploit them to *unfold* the interpretation into the canonical interpretation of Π and $\mathcal{D}$.

For convenience of presentation, we fix for the remainder of this subsection an arbitrary bounded program Π, bounded dataset $\mathcal{D}$, and natural number k. To make the first observation we need to introduce additional notions. We let the *depth* of a rule r, written as $\mathsf{depth}(r)$, be the sum of the right endpoints in all intervals occurring in r (or 0 if r mentions no intervals). For example, the depth of Rule (3) is $183 + 10 = 193$. The depth of a program Π, written as $\mathsf{depth}(\Pi)$, is the maximum depth of its rules; hence, bounded programs have finite depth. Moreover, for an interpretation $\mathfrak{I}$ and an interval ϱ, we let the *projection* $\mathfrak{I}\,|_{\varrho}$ of $\mathfrak{I}$ over ϱ be the interpretation that coincides with $\mathfrak{I}$ on ϱ and makes all relational atoms false outside ϱ.

As shown in the following proposition, the depth of a bounded rule determines the time points that can be "affected" by an application of this rule.

▶ **Proposition 17.** *For every interpretation $\mathfrak{I}$, time point t, and bounded rule r, it holds that $\mathfrak{I}$ satisfies r at t if and only if so does $\mathfrak{I}\,|_{[t-\mathsf{depth}(r),t+\mathsf{depth}(r)]}$.*

Proof. By definition, $\mathfrak{I}$ satisfies r at t if and only if the body of r does not hold at t or the head of r holds at t. By the definition of $\mathsf{depth}(r)$, all the facts corresponding to the satisfaction of the body and the head of r at t are over intervals contained in $[t-\mathsf{depth}(r), t+\mathsf{depth}(r)]$. This implies the claim in the proposition. ◀

Our next aim is to define a saturated interpretation. For this, however, we will need to introduce additional concepts.

For a dataset $\mathcal{D}$, we let $t^-_{\mathcal{D}}$ and $t^+_{\mathcal{D}}$ be the minimal and maximal numbers mentioned as interval endpoints in $\mathcal{D}$ (if $\mathcal{D}$ does not mention any numbers, we let both $t^-_{\mathcal{D}}$ and $t^+_{\mathcal{D}}$ be 0). We say that an interpretation $\mathfrak{I}$ satisfies Π in an interval ϱ if, for each $r \in \mathsf{ground}(\Pi)$ and each $t \in \varrho$, whenever $\mathfrak{I}$ satisfies each body atom of r at t, then $\mathfrak{I}$ satisfies the head of r at t. Furthermore, we say that an interpretation $\mathfrak{I}'$ is a *shift* of $\mathfrak{I}$ if there exists a rational number q such that $\mathfrak{I} \models M@\varrho$ if and only if $\mathfrak{I}' \models M@(\varrho + q)$, for each fact $M@\varrho$.

With the above concepts introduced, we are ready to present the definition of a saturated interpretation.

▶ **Definition 18.** *Interpretation $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ is* saturated *if there exist closed intervals ϱ_1, ϱ_2, ϱ_3, and ϱ_4 of length $2 \cdot \mathsf{depth}(\Pi)$, whose endpoints are located on the $(\Pi, \mathcal{D})$-ruler and satisfy $\varrho_1^+ < \varrho_2^+ < t^-_{\mathcal{D}}$ and $t^+_{\mathcal{D}} < \varrho_3^- < \varrho_4^-$, and such that the following properties hold:*

- *$T^k_\Pi(\mathfrak{I}_\mathcal{D})$ satisfies Π in $[\varrho_1^-, \varrho_4^+]$;*
- *$T^k_\Pi(\mathfrak{I}_\mathcal{D})\,|_{\varrho_1}$ and $T^k_\Pi(\mathfrak{I}_\mathcal{D})\,|_{\varrho_3}$ are shifts of $T^k_\Pi(\mathfrak{I}_\mathcal{D})\,|_{\varrho_2}$ and $T^k_\Pi(\mathfrak{I}_\mathcal{D})\,|_{\varrho_4}$, respectively.*

Any pair of intervals $[\varrho_1^-, \varrho_2^-)$ and $(\varrho_3^+, \varrho_4^+]$, for ϱ_1, ϱ_2, ϱ_3, ϱ_4 as above, will be referred to as periods *of $T^k_\Pi(\mathfrak{I}_\mathcal{D})$.*

Intuitively, a saturated interpretation contains a "central fragment" $[\varrho_1^-, \varrho_4^+]$ which satisfies $\mathcal{D}$ and such that a single application of T_Π does not derive any new facts within this fragment (i.e., the interpretation satisfies Π within $[\varrho_1^-, \varrho_4^+]$). In the left segment of this "central fragment" there are two intervals ϱ_1 and ϱ_2 – each of length $2 \cdot \mathsf{depth}(\Pi)$ – in which $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ satisfies the same relational facts modulo a shift. Analogously, in the right segment of the "central fragment" there are intervals ϱ_3 and ϱ_4 also with repeating contents.

It is worth emphasising that in the definition of a saturated interpretation (first item), we consider only a single materialisation step, which can be effectively checked. As we will show in Theorem 21, rather surprisingly, this condition is enough to guarantee that a saturated interpretation can be unfolded into the canonical interpretation.

We next define the unfolding of a saturated interpretation relatively to a pair $(\varrho_{\mathsf{left}}, \varrho_{\mathsf{right}})$ of its periods. Although there can be many such pairs of intervals, the key properties of the unfolding are independent of the choice of periods.

▶ **Definition 19.** *The $(\varrho_{\mathsf{left}}, \varrho_{\mathsf{right}})$-unfolding, of a saturated interpretation $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ with periods $(\varrho_{\mathsf{left}}, \varrho_{\mathsf{right}})$, is the interpretation $\mathfrak{C}$ such that:*

- $\mathfrak{C}\,|_{[\varrho^-_{\mathsf{left}}, \varrho^+_{\mathsf{right}}]} = T^k_\Pi(\mathfrak{I}_\mathcal{D})\,|_{[\varrho^-_{\mathsf{left}}, \varrho^+_{\mathsf{right}}]}$,
- $\mathfrak{C}\,|_{\varrho_{\mathsf{left}} - n\cdot|\varrho_{\mathsf{left}}|}$ *is a shift of* $T^k_\Pi(\mathfrak{I}_\mathcal{D})\,|_{\varrho_{\mathsf{left}}}$, *for any* $n \in \mathbb{N}$,
- $\mathfrak{C}\,|_{\varrho_{\mathsf{right}} + n\cdot|\varrho_{\mathsf{right}}|}$ *is a shift of* $T^k_\Pi(\mathfrak{I}_\mathcal{D})\,|_{\varrho_{\mathsf{right}}}$, *for any* $n \in \mathbb{N}$.

We observe that the unfolding is unique and can be obtained from the least interpretation coinciding with $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ on $[\varrho^-_{\mathsf{left}}, \varrho^+_{\mathsf{right}}]$ by subsequently copying the fragment of this interpretation that spans ϱ_{left} infinitely many times to the left and the fragment of this interpretation that spans ϱ_{right} infinitely many times towards the right on the timeline.

In Theorem 21 we will show that the unfolding of a saturated interpretation coincides with the canonical interpretation. The proof of this theorem exploits the following technical lemma.

▶ **Lemma 20.** *Let ϱ be a closed interval of length $\mathsf{depth}(\Pi)$ and $\mathcal{D}'$ a dataset representing $\mathfrak{C}_{\Pi,\mathcal{D}}|_\varrho$. Then both of the following hold:*

1. *If $\varrho^+ \geq t^+_\mathcal{D}$, then $\mathfrak{C}_{\Pi,\mathcal{D}}|_{(\varrho^+,\infty)} = \mathfrak{C}_{\Pi,\mathcal{D}'}|_{(\varrho^+,\infty)}$.*
2. *If $\varrho^- \leq t^-_\mathcal{D}$, then $\mathfrak{C}_{\Pi,\mathcal{D}}|_{(-\infty,\varrho^-)} = \mathfrak{C}_{\Pi,\mathcal{D}'}|_{(-\infty,\varrho^-)}$.*

Proof sketch. Both items have similar proofs, so we focus on the first one. The inclusion $\mathfrak{C}_{\Pi,\mathcal{D}'}|_{(\varrho^+,\infty)} \subseteq \mathfrak{C}_{\Pi,\mathcal{D}}|_{(\varrho^+,\infty)}$ follows from the fact that $\mathfrak{C}_{\Pi,\mathcal{D}} \models \mathcal{D}'$, so $\mathfrak{C}_{\Pi,\mathcal{D}'} \subseteq \mathfrak{C}_{\Pi,\mathcal{D}}$. To show that $\mathfrak{C}_{\Pi,\mathcal{D}}|_{(\varrho^+,\infty)} \subseteq \mathfrak{C}_{\Pi,\mathcal{D}'}|_{(\varrho^+,\infty)}$ it suffices to prove inductively that, for every ordinal α and relational fact $M@t$ with $t > \varrho^+$, if $T^\alpha_\Pi(\mathfrak{I}_\mathcal{D}) \models M@t$, then $T^\alpha_\Pi(\mathfrak{I}_{\mathcal{D}'}) \models M@t$. ◀

▶ **Theorem 21.** *The $(\varrho_{\mathsf{left}}, \varrho_{\mathsf{right}})$-unfolding $\mathfrak{C}$ of a saturated interpretation $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ with periods $(\varrho_{\mathsf{left}}, \varrho_{\mathsf{right}})$ coincides with the canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$.*

Proof sketch. We first show that $\mathfrak{C}_{\Pi,\mathcal{D}} \subseteq \mathfrak{C}$. Since $\mathfrak{C}_{\Pi,\mathcal{D}}$ is the least model of Π and $\mathcal{D}$, it suffices to show that $\mathfrak{C}$ is a model of Π and $\mathcal{D}$. As $[t^-_\mathcal{D}, t^+_\mathcal{D}] \subseteq [\varrho^-_{\mathsf{left}}, \varrho^+_{\mathsf{right}}]$, we obtain that the projection of $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ over $[\varrho^-_{\mathsf{left}}, \varrho^+_{\mathsf{right}}]$ is a model of $\mathcal{D}$. However, by Definition 19, $\mathfrak{C}$ and $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ coincide on $[\varrho^-_{\mathsf{left}}, \varrho^+_{\mathsf{right}}]$, so $\mathfrak{C}$ is also a model of $\mathcal{D}$. To show that $\mathfrak{C}$ satisfies Π in $\varrho = [\varrho^-_{\mathsf{left}} + \mathsf{depth}(\Pi), \varrho^+_{\mathsf{right}} - \mathsf{depth}(\Pi)]$, it suffices, by Proposition 17, to show that the projection of $\mathfrak{C}$ over $[\varrho^-_{\mathsf{left}}, \varrho^+_{\mathsf{right}}]$ satisfies Π in ϱ; this holds since $\mathfrak{C}$ and $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ coincide on $[\varrho^-_{\mathsf{left}}, \varrho^+_{\mathsf{right}}]$ and $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ satisfies Π in $[\varrho^-_{\mathsf{left}}, \varrho^+_{\mathsf{right}}]$, as it is saturated. It remains to argue that $\mathfrak{C}$ satisfies Π at each t outside ϱ. Assume that $t < \varrho^-$, so $t < \varrho^-_{\mathsf{left}} + \mathsf{depth}(\Pi)$. Then, there exists a unique pair of $n \in \mathbb{N}$ and $t' \in [\varrho^-_{\mathsf{left}} + \mathsf{depth}(\Pi), \varrho^+_{\mathsf{left}} + \mathsf{depth}(\Pi))$ such that $t + n \cdot |\varrho_{\mathsf{left}}| = t'$. This, by the definition of $\mathfrak{C}$, implies that $\mathfrak{C}|_{[t-\mathsf{depth}(\Pi), t+\mathsf{depth}(\Pi)]}$ is a shift of $\mathfrak{C}|_{[t'-\mathsf{depth}(\Pi), t'+\mathsf{depth}(\Pi)]}$. Thus, by Proposition 17, $\mathfrak{C}$ satisfies Π at t if and only if $\mathfrak{C}$ satisfies Π at t'. The latter holds since $t' \in \varrho$ and, as we already showed, $\mathfrak{C}$ satisfies Π in ϱ. The proof for $t > \varrho^-_{\mathsf{right}} - \mathsf{depth}(\Pi)$ is similar, so $\mathfrak{C}_{\Pi,\mathcal{D}} \subseteq \mathfrak{C}$.

We show that $\mathfrak{C} \subseteq \mathfrak{C}_{\Pi,\mathcal{D}}$. The projection of $\mathfrak{C}$ over $[\varrho^-_{\mathsf{left}}, \varrho^+_{\mathsf{right}}]$ is in $\mathfrak{C}_{\Pi,\mathcal{D}}$ since $\mathfrak{C}$ and $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ coincide on this interval and $T^k_\Pi(\mathfrak{I}_\mathcal{D}) \subseteq \mathfrak{C}_{\Pi,\mathcal{D}}$. To show that the projection of $\mathfrak{C}$ over $(-\infty, \varrho^-_{\mathsf{left}}]$ is in $\mathfrak{C}_{\Pi,\mathcal{D}}$, we argue that $\mathfrak{C}|_{[t_i, \varrho^+_{\mathsf{right}}]} \subseteq \mathfrak{C}_{\Pi,\mathcal{D}}|_{[t_i, \varrho^+_{\mathsf{right}}]}$ by induction on consecutive timepoints t_i on the $(\Pi, \mathcal{D})$-ruler, with $t_0 = \varrho^-_{\mathsf{left}}$. The base holds since $\mathfrak{C}$ and $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ coincide on $[\varrho^-_{\mathsf{left}}, \varrho^+_{\mathsf{right}}]$. For the inductive step, let $\varrho_A = [t_i, t_{i-1})$, $\varrho_B = [t_{i-1}, t_{i-1} + \mathsf{depth}(\Pi)]$, $\varrho'_A = \varrho_A + |\varrho_{\mathsf{left}}|$, and $\varrho'_B = \varrho_B + |\varrho_{\mathsf{left}}|$. Hence, it suffices to show that $\mathfrak{C}|_{\varrho_A} \subseteq \mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho_A}$. We note that $\mathfrak{C}|_{\varrho_B}$ is a shift of $\mathfrak{C}|_{\varrho'_B}$ by the construction of $\mathfrak{C}$ and the fact that $(\varrho'_B)^+ \leq \varrho^+_{\mathsf{left}} + \mathsf{depth}(\Pi)$. Thus, by the inductive assumption and $\mathfrak{C}_{\Pi,\mathcal{D}} \subseteq \mathfrak{C}$, we obtain that $\mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho_B}$ is a shift of $\mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho'_B}$. By the construction of $\mathfrak{C}$ we know that $\mathfrak{C}|_{\varrho_A}$ is a shift of $\mathfrak{C}|_{\varrho'_A}$ and since the lengths of ϱ_B and ϱ'_B equal $\mathsf{depth}(\Pi)$, we can use Lemma 20 to show that $\mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho_A}$ is a shift of $\mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho'_A}$. By the inductive assumption, $\mathfrak{C}|_{\varrho_{A'}} \subseteq \mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho_{A'}}$ and so, $\mathfrak{C}|_{\varrho_A} \subseteq \mathfrak{C}_{\Pi,\mathcal{D}}|_{\varrho_A}$. The proof of the inclusion $\mathfrak{C}|_{[\varrho^+_{\mathsf{right}}, \infty)} \subseteq \mathfrak{C}_{\Pi,\mathcal{D}}|_{[\varrho^+_{\mathsf{right}}, \infty)}$ is symmetric; thus, $\mathfrak{C} \subseteq \mathfrak{C}_{\Pi,\mathcal{D}}$. ◀

Theorem 21 states that the unfolding of a saturated interpretation around any pair of its periods coincides with the canonical interpretation. Thus, we can refer to an unfolding of a saturated interpretation without explicitly referring to its periods.

We conclude this section by establishing a bound k_{max}, depending on Π and $\mathcal{D}$, which ensures that $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ is saturated for some $k \leq k_{\mathsf{max}}$. Intuitively, k_{max} is the number of facts that we can construct by combining atoms relevant to Π and $\mathcal{D}$ with $(\Pi, \mathcal{D})$-intervals contained in a sufficiently large interval ϱ. The interval ϱ is chosen so that $[t^-_\mathcal{D}, t^+_\mathcal{D}] \subseteq \varrho$, and such that its fragments $[\varrho^-, t^-_\mathcal{D})$ and $(t^+_\mathcal{D}, \varrho^+]$ to the left and right of $\mathcal{D}$, respectively, contain so many intervals of length $2 \cdot \mathsf{depth}(\Pi)$ with endpoints on the $(\Pi, \mathcal{D})$-ruler, that their contents need to repeat. These repetitions guarantee the existence of intervals $\varrho_1, \dots, \varrho_4$ from Definition 18.

▶ **Theorem 22.** *There exists $k \leq k_{\mathsf{max}}$ such that $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ is saturated, where the bound k_{max} is as follows.*

Let A be the number of ground relational atoms in the grounding of Π with constants from Π and $\mathcal{D}$, let B be the number of $(\Pi, \mathcal{D})$-intervals within $[t^-_\mathcal{D}, t^-_\mathcal{D} + 2 \cdot \mathsf{depth}(\Pi)]$, and let $\varrho = [t_{-w} - 2 \cdot \mathsf{depth}(\Pi), t_w + 2 \cdot \mathsf{depth}(\Pi)]$, where $\dots < t_{-2} < t_{-1} < t^-_\mathcal{D}$ and $t^+_\mathcal{D} < t_1 < t_2 < \dots$ are sequences of consecutive time points on the $(\Pi, \mathcal{D})$-ruler, while $w = 1 + B \cdot (2^A)^B$. Then k_{max} is the product of A and the number of $(\Pi, \mathcal{D})$-intervals contained in ϱ.

Proof sketch. The canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$ satisfies at most k_{max} relational facts over $(\Pi, \mathcal{D})$-intervals contained in ϱ. Since each application of T_Π (before reaching a fixpoint) introduces at least one new relational fact over a $(\Pi, \mathcal{D})$-interval, there needs to exist $k \leq k_{\mathsf{max}}$ such that $T^k_\Pi(\mathfrak{I}_\mathcal{D}) \mid_\varrho = T^{k+1}_\Pi(\mathfrak{I}_\mathcal{D}) \mid_\varrho$. We argue that $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ is saturated for such k by showing that ϱ contains intervals $\varrho_1, \dots, \varrho_4$ satisfying the conditions from Definition 18. To show the existence of ϱ_1 and ϱ_2, we observe that ϱ contains $w = 1 + B \cdot (2^A)^B$ intervals of length $2 \cdot \mathsf{depth}(\Pi)$ whose endpoints are located on the $(\Pi, \mathcal{D})$-ruler to the left of $t^-_\mathcal{D}$. We can show that there must exist amongst them two distinct intervals ϱ_1 and ϱ_2 such that $T^k_\Pi(\mathfrak{I}_\mathcal{D}) \mid_{\varrho_1}$ is a shift of $T^k_\Pi(\mathfrak{I}_\mathcal{D}) \mid_{\varrho_2}$. Indeed, each interval of the form $[t, t + 2 \cdot \mathsf{depth}(\Pi)]$, for t on the $(\Pi, \mathcal{D})$-ruler, contains the same number of $(\Pi, \mathcal{D})$-intervals as $[t^-_\mathcal{D}, t^-_\mathcal{D} + 2 \cdot \mathsf{depth}(\Pi)]$ does, which equals B. In each of these $(\Pi, \mathcal{D})$-intervals there can hold at most 2^A combinations of relational atoms, which gives rise to $(2^A)^B$ different contents of an interval of the form $[t, t + 2 \cdot \mathsf{depth}(\Pi)]$. Additionally, these intervals can differ depending on the location of $(\Pi, \mathcal{D})$-intervals they contain. By the definition of the $(\Pi, \mathcal{D})$-ruler, there are at most B different layouts of $(\Pi, \mathcal{D})$-intervals contained in an interval of the form $[t, t + 2 \cdot \mathsf{depth}(\Pi)]$. Hence, in total, there are at most $B \cdot (2^A)^B$ intervals of the form $[t, t + 2 \cdot \mathsf{depth}(\Pi)]$ with different contents. Thus, in a set of $w = 1 + B \cdot (2^A)^B$ such intervals, there needs to be a pair of distinct intervals ϱ_1 and ϱ_2 such that $T^k_\Pi(\mathfrak{I}_\mathcal{D}) \mid_{\varrho_1}$ is a shift of $T^k_\Pi(\mathfrak{I}_\mathcal{D}) \mid_{\varrho_2}$. Analogously, we can show the existence of required ϱ_3 and ϱ_4 to the right of $t^+_\mathcal{D}$, so the second item from Definition 18 holds. Moreover, since $[\varrho^-_1, \varrho^+_4] \subseteq \varrho$ and $T^k_\Pi(\mathfrak{I}_\mathcal{D}) \mid_\varrho = T^{k+1}_\Pi(\mathfrak{I}_\mathcal{D}) \mid_\varrho$, we obtain that $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ satisfies Π in $[\varrho^-_1, \varrho^+_4]$. Thus, the first item from Definition 18 holds as well, and so, $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ is saturated. ◀

The bound k_{max} from Theorem 22 is doubly exponential in the size of Π and $\mathcal{D}$, but only exponential in the size of $\mathcal{D}$. This bound shows us how much time is needed to perform reasoning, which is consistent with the computational complexity of reasoning in DatalogMTL, namely ExpSpace in combined complexity [11] and PSpace in data complexity [46].

In the next section we will exploit the obtained results to provide a terminating reasoning algorithm.

4.3 Reasoning Algorithm Based on Saturation

The results from the previous section suggest the reasoning procedure formalised in Algorithm 3, which checks if a bounded program Π and a dataset $\mathcal{D}$ entail a fact $M@\varrho$.

Algorithm 3 Reasoning via Periods Detection.

Input: a bounded program Π, a bounded dataset $\mathcal{D}$, and a fact $M@\varrho$
Output: a Boolean value True or False

1 $\mathcal{D}_{\mathsf{now}} \coloneqq \mathcal{D}$;
2 **loop**
3 **if** $\mathcal{D}_{\mathsf{now}} \models M@\varrho$ **then return** True;
4 $(\varrho_{\mathsf{left}}, \varrho_{\mathsf{right}}) \coloneqq \mathsf{Periods}(\Pi, \mathcal{D}, \mathcal{D}_{\mathsf{now}})$;
5 **if** $\varrho_{\mathsf{left}} \neq \emptyset$ *and* $\varrho_{\mathsf{right}} \neq \emptyset$ **then**
6 **if** $\mathsf{Entails}(\Pi, \mathcal{D}, \mathcal{D}_{\mathsf{now}}, \varrho_{\mathsf{left}}, \varrho_{\mathsf{right}}, M@\varrho)$ **then return** True;
7 **else return** False;
8 $\mathcal{D}_{\mathsf{now}} \coloneqq \mathsf{ApplyRules}(\Pi, \mathcal{D}_{\mathsf{now}})$;

The algorithm successively applies the rules of Π to the dataset (Line 8) until one of two stopping conditions hold. The first condition (Line 3) detects if the materialisation $\mathcal{D}_{\mathsf{now}}$ constructed thus far entails the input fact $M@\varrho$, in which case the algorithm reports that the input fact is entailed. The second condition checks if $\mathcal{D}_{\mathsf{now}}$ is saturated (i.e., represents a saturated interpretation). To this end, the algorithm computes in Line 4 two non-empty intervals which are periods of $\mathcal{D}_{\mathsf{now}}$ (if $\mathcal{D}_{\mathsf{now}}$ is saturated), or introduces two empty intervals (if $\mathcal{D}_{\mathsf{now}}$ is not saturated). If $\mathcal{D}_{\mathsf{now}}$ is saturated, the algorithm exploits $\mathcal{D}_{\mathsf{now}}$ and its periods to check whether $M@\varrho$ is entailed (Lines 6–7).

We next describe in detail the computations of the algorithm and establish its correctness. In what follows, we fix an arbitrary input Π, $\mathcal{D}$, and $M@\varrho$ to Algorithm 3 and assume that the notions of a saturated interpretation and its periods are relative to Π and $\mathcal{D}$. We also let the projection $\mathcal{D}' \restriction_{\varrho'}$ of a dataset $\mathcal{D}'$ over an interval ϱ' be the dataset obtained by intersecting all intervals in facts from $\mathcal{D}'$ with ϱ'.

Rule Application. In each iteration of the loop, Algorithm 3 applies the procedure $\mathsf{ApplyRules}$ (Line 8), which implements a single round of rule applications to $\mathcal{D}_{\mathsf{now}}$, and thus, mimics an application of the immediate consequence operator T_Π to $\mathfrak{I}_{\mathcal{D}_{\mathsf{now}}}$. Hence, in the beginning of a $(k+1)$st iteration of the loop, the dataset $\mathcal{D}_{\mathsf{now}}$ in Algorithm 3 represents $T^k_\Pi(\mathfrak{I}_\mathcal{D})$.

First Stopping Condition. In Line 3, Algorithm 3 checks whether the materialisation $\mathcal{D}_{\mathsf{now}}$ constructed so far entails the input fact $M@\varrho$. Since facts in $\mathcal{D}_{\mathsf{now}}$ are stored in a coalesced form, to check if $\mathcal{D}_{\mathsf{now}} \models M@\varrho$, it suffices to scan $\mathcal{D}_{\mathsf{now}}$ and verify whether there is $M@\varrho' \in \mathcal{D}_{\mathsf{now}}$ with $\varrho \subseteq \varrho'$.

Second Stopping Condition. Algorithm 3 calls the $\mathsf{Periods}$ procedure in Line 4 to check, in an iteration $k+1$ of the loop, if $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ is saturated. The procedure searches for intervals $\varrho_1, \dots, \varrho_4$ satisfying the conditions in Definition 18 and returns the periods of the interpretation $T^k_\Pi(\mathfrak{I}_\mathcal{D})$ if it is saturated or a pair of empty intervals otherwise. To this end, the procedure performs one round of rule applications to $\mathcal{D}_{\mathsf{now}}$, computing $\mathcal{D}_{\mathsf{next}}$. Then, it computes an interval ϱ_{max} as either the empty interval (if $\mathcal{D}_{\mathsf{now}}$ and $\mathcal{D}_{\mathsf{next}}$ do not coincide on $[t^-_\mathcal{D}, t^+_\mathcal{D}]$), or otherwise as the maximal interval containing $[t^-_\mathcal{D}, t^+_\mathcal{D}]$ and such that $\mathcal{D}_{\mathsf{now}}$ and $\mathcal{D}_{\mathsf{next}}$ coincide on ϱ_{max}. Interval ϱ_{max} is next used to search for ϱ_1 and ϱ_2, namely all pairs of intervals contained in ϱ_{max}, located to the left of $t^-_\mathcal{D}$, with endpoints on the $(\Pi, \mathcal{D})$-ruler, and of lengths $2 \cdot \mathsf{depth}(\Pi)$, are compared. The first pair of such intervals with the same

contents in $\mathcal{D}_{\mathsf{now}}$ is set as ϱ_1 and ϱ_2; otherwise ϱ_1 and ϱ_2 are empty intervals. In a similar way intervals ϱ_3 and ϱ_4 are computed. Finally, the procedure outputs a pair of intervals $([\varrho_1^-, \varrho_2^-), (\varrho_3^+, \varrho_4^+])$, or $(\emptyset, \emptyset)$ if any of the intervals $\varrho_1, \dots, \varrho_4$ is empty.

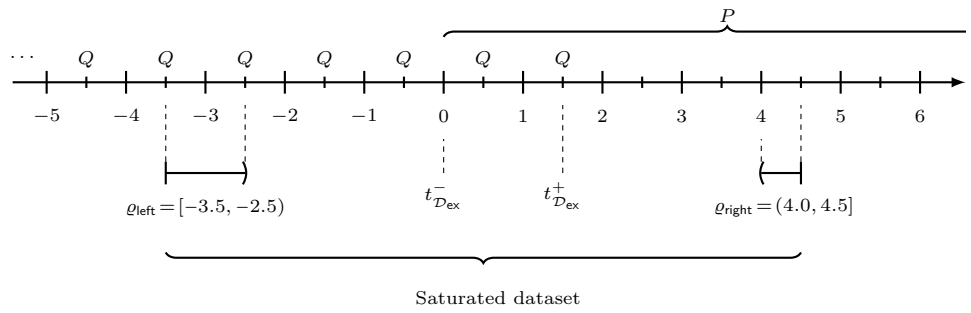
Fact Entailment Checking after Saturation. After constructing a saturated dataset $\mathcal{D}_{\mathsf{now}}$ (representing $T_\Pi^k(\mathfrak{I}_\mathcal{D})$, for some $k \in \mathbb{N}$) with periods $(\varrho_{\mathsf{left}}, \varrho_{\mathsf{right}})$, Algorithm 3 calls the procedure Entails (Line 6) to check whether the input fact $M@\varrho$ holds in the unfolding of $T_\Pi^k(\mathfrak{I}_\mathcal{D})$. This can be easily checked by exploiting Definition 19 of unfolding.

From our theoretical results in the previous subsection it follows that the algorithm is sound and complete:

▶ **Theorem 23.** *Algorithm 3 outputs* True *if* $(\Pi, \mathcal{D}) \models M@\varrho$*, otherwise it outputs* False*. Moreover, the algorithm terminates after at most* $k_{\mathsf{max}} + 1$ *(c.f. Theorem 22) iterations of its main loop.*

We conclude this section with an example illustrating the execution of Algorithm 3.

▶ **Example 24.** Consider $\Pi = \{\boxplus_{[0,1]} P \leftarrow P, \boxminus_{[1,1]} Q \leftarrow Q\}$, $\mathcal{D} = \{P@0, Q@1.5\}$, and a query fact $M@t = Q@{-4.5}$. After 5 iterations of the loop in Algorithm 3, the dataset $\mathcal{D}_{\mathsf{now}}$ consists of facts $P@[0,5]$ and $Q@t$, for all $t \in \{-3.5, -2.5, -1.5, -0.5, 0.5, 1.5\}$. The stopping condition from Line 3 does not hold, but the condition in Line 5 does. Indeed, $\mathcal{D}_{\mathsf{now}}$ is saturated and its periods computed in Line 4 are $\varrho_{\mathsf{left}} = [-3.5, -2.5)$ and $\varrho_{\mathsf{right}} = (4.0, 4.5]$. The unfolding of $\mathcal{D}_{\mathsf{now}}$ is the canonical interpretation of Π and $\mathcal{D}$ depicted in the figure below. Then, in Line 6, Algorithm 3 can exploit $\mathcal{D}_{\mathsf{now}}$, ϱ_{left}, and ϱ_{right} to detect entailment of any fact. In particular, the input fact $Q@{-4.5}$ is entailed, and so, Algorithm 3 returns True in Line 6.



5 Time to Conclude

We have introduced DatalogMTL, as an expressive rule-based language for temporal reasoning. After presenting formal syntax and semantics of DatalogMTL, we have discussed its reasoning algorithms. In particular, we have discussed two ways to perform reasoning. The first is based on iterative application of rules of a DatalogMTL program until no further facts can be derived. Such a procedure can be easily implemented, but it does not guarantee termination. Then, we have presented an algorithm which extends materialisation with checking an additional condition – which we call saturation. A saturated partial materialisation can always be constructed in a bounded number of steps, and can be used to construct the full canonical interpretation.

In what follows we will present a list of further research topics and papers on DatalogMTL, as well as some interesting open problems.

5.1 Further Reading and Related Work

DatalogMTL was first introduced under the continuous semantics and over the rational timeline [11]. The complexity of standard reasoning tasks in DatalogMTL and its fragments was further investigated [12, 46, 48, 8]. DatalogMTL has also been studied over the integer timeline [47] and under the event-based semantics [36]. DatalogMTL has recently been extended with negation-as-failure under the stable model semantics, first for stratified programs [38] and subsequently for the general case [52]. It was also studied in the setting of existential rules [30]. Very recently, inconsistency handling in DatalogMTL has been studied, by introducing temporal versions of conflicts and repairs, as well as establishing the computational complexity of their generation [10].

MeTeoR [41] 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 [37]. It has been further optimised with several techniques such as seminaive materialisation [40, 42] and magic sets [45]. Earlier systems with DatalogMTL support, such as the Ontop platform [28], performed reasoning via rewriting into SQL and were restricted to non-recursive programs. Reasoning methods based on solving linear inequalities for a language similar to DatalogMTL were also introduced [14]. Another reasoning approach has been recently introduced by extending the Datalog system Vadalog [7]. DatalogMTL has proved useful for temporal stream reasoning [49, 37, 50], temporal ontology-based data access [11], specification and verification of banking agreements [34], fact-checking economic claims [31], and for describing dance movements [35], among others. DatalogMTL has been also studied in the context of learning temporal rules from datasets [43] and to study the ability of LLMs to perform temporal reasoning [20, 44].

Extensions of DatalogMTL with non-monotonic negation are closely related to temporal extensions of answer set programming (ASP) [2]. Particularly relevant is a recently introduced extension of ASP with MTL operators [16]. Previously, ASP was also extended with LTL operators, giving rise to temporal equilibrium logic TEL [18, 23, 15], which has been implemented in the Telingo system [17]. Additionally, LARS [6] 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 [24, 25]. Operators from MTL have also been exploited in temporal extensions of description logics (DLs) [26, 5, 39, 4].

5.2 Open Problems

Research on DatalogMTL has opened a number of interesting directions and left several important questions unanswered. We highlight some of them below:

- **Algorithms:** The saturation-based algorithm presented in Section 4 is designed only for the bounded setting, namely for programs and datasets that do not use ∞ or $-\infty$ as interval endpoints. Extending this approach to the full language of DatalogMTL remains an open problem. Such an extension appears to require not just the computation of a single saturated interpretation, but rather a sequence of them. Developing such an algorithm and establishing its correctness constitutes a significant technical challenge of high theoretical and practical importance.
- **Computational Complexity**: A number of tight complexity bounds for DatalogMTL and its fragments are already known. However, one fragment whose complexity remains

unresolved – despite substantial effort – is DatalogMTL$^{\boxminus}_{\mathsf{core}}$, which allows only core rules (rules with head $\bot$ or at most one body atom) and restricts temporal operators to $\boxminus$. Interestingly, similar "core box" fragments also lack tight complexity results in other temporal logics, such as interval temporal logics. This suggests the existence of a broader complexity challenge. Progress on DatalogMTL could provide insights that extend beyond this language, potentially unlocking solutions to analogous problems in other temporal frameworks.

- **Characterisation of Neural Models**: Recent work has established a close correspondence between the expressive power of Datalog and Graph Neural Networks (GNNs) – neural architectures designed for processing graph-structured data [21]. In parallel, temporal extensions of GNNs have emerged [51]. This naturally raises the question of whether the expressive power of such temporal architectures can be captured by temporal extensions of Datalog, such as DatalogMTL.

The open problems we have listed illustrate just some of the many directions in which the study of DatalogMTL can advance. They range from fundamental algorithmic challenges, through unresolved complexity questions, to emerging connections with neural models. This list is by no means exhaustive, and together these questions highlight exciting opportunities for future research.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

2 Felicidad Aguado, Pedro Cabalar, Martín Diéguez, Gilberto Pérez, Torsten Schaub, Anna Schuhmann, and Concepción Vidal. Linear-time temporal answer set programming. *Theory and Practice of Logic Programming*, pages 1–55, 2021.

3 Wałęga Przemysław Andrzej, Michał Zawidzki, and Christoph Haase. Computing all facts entailed by an ltl specification. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, pages 679–689, 2023.

4 Alessandro Artale and Enrico Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research*, 9:463–506, 1998. doi:10.1613/JAIR.516.

5 Franz Baader, Stefan Borgwardt, Patrick Koopmann, Ana Ozaki, and Veronika Thost. Metric temporal description logics with interval-rigid names. *ACM Trans. Comput. Log.*, 21(4):30:1–30:46, 2020. doi:10.1145/3399443.

6 Harald Beck, Minh Dao-Tran, Thomas Eiter, and Christian Folie. Stream reasoning with LARS. *KI - Künstliche Intelligenz*, 32(2):193–195, 2018. doi:10.1007/S13218-018-0537-9.

7 Luigi Bellomarini, Livia Blasi, Markus Nissl, and Emanuel Sallinger. The temporal vadalog system: Temporal datalog-based reasoning. *Theory and Practice of Logic Programming*, 25(2):168–196, 2025. doi:10.1017/S1471068425000018.

8 Luigi Bellomarini, Markus Nissl, and Emanuel Sallinger. Query evaluation in DatalogMTL – taming infinite query results. *CoRR*, abs/2109.10691, 2021. arXiv:2109.10691.

9 Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. The Vadalog system: Datalog-based reasoning for knowledge graphs. *Proceedings of the VLDB Endowment*, 11(9):975–987, 2018. doi:10.14778/3213880.3213888.

10 Meghyn Bienvenu, Camille Bourgaux, and Atefe Khodadaditaghanaki. Inconsistency handling in datalogmtl. *arXiv preprint arXiv:2505.10394*, 2025. doi:10.48550/arXiv.2505.10394.

11 Sebastian Brandt, Elem Güzel Kalayci, Roman Kontchakov, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. Ontology-based data access with a Horn fragment of metric temporal logic. In *Proc. of AAAI*, pages 1070–1076, 2017. doi:10.1609/AAAI.V31I1.10696.

12 Sebastian Brandt, Elem Güzel Kalaycı, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. Querying log data with metric temporal logic. *J. Artif. Intell. Res.*, 62:829–877, 2018. doi:10.1613/JAIR.1.11229.

13 François Bry, Norbert Eisinger, Thomas Eiter, Tim Furche, Georg Gottlob, Clemens Ley, Benedikt Linse, Reinhard Pichler, and Fang Wei. Foundations of rule-based query answering. In *Reasoning Web*, volume 4636, pages 1–153, 2007. doi:10.1007/978-3-540-74615-7_1.

14 Christoph Brzoska. Programming in metric temporal logic. *Theoretical Computer Science*, 202(1-2):55–125, 1998. doi:10.1016/S0304-3975(97)00139-4.

15 Pedro Cabalar. Temporal ASP: From logical foundations to practical use with telingo. In Mantas Šimkus and Ivan Varzinczak, editors, *Reasoning Web. Declarative Artificial Intelligence*, pages 94–114, 2022.

16 Pedro Cabalar, Martín Diéguez, Torsten Schaub, and Anna Schuhmann. Towards metric temporal answer set programming. *Theory and Practice of Logic Programming*, 20(5):783–798, 2020. doi:10.1017/S1471068420000307.

17 Pedro Cabalar, Roland Kaminski, Philip Morkisch, and Torsten Schaub. telingo = asp + time. In Marcello Balduccini, Yuliya Lierler, and Stefan Woltran, editors, *Logic Programming and Nonmonotonic Reasoning*, pages 256–269, 2019. doi:10.1007/978-3-030-20528-7_19.

18 Pedro Cabalar and Gilberto Pérez Vega. Temporal equilibrium logic: A first approach. In Roberto Moreno Díaz, Franz Pichler, and Alexis Quesada Arencibia, editors, *Computer Aided Systems Theory*, pages 241–248, 2007. doi:10.1007/978-3-540-75867-9_31.

19 David Carral, Irina Dragoste, Larry González, Ceriel J. H. Jacobs, Markus Krötzsch, and Jacopo Urbani. Vlog: A rule engine for knowledge graphs. In *The Semantic Web - ISWC*, volume 11779, pages 19–35, 2019. doi:10.1007/978-3-030-30796-7_2.

20 Andrea Colombo, Teodoro Baldazzi, Luigi Bellomarini, Andrea Gentili, and Emanuel Sallinger. Llm-based datalogmtl modelling of micar-compliant crypto-assets markets. In *Datalog in Academia and Industry (Datalog-2.0 2024)*, volume 3801, pages 17–22, 2024. URL: https://ceur-ws.org/Vol-3801/short1.pdf.

21 David Jaime Tena Cucala, Bernardo Cuenca Grau, Egor V. Kostylev, and Boris Motik. Explainable GNN-based models over knowledge graphs. In *International Conference on Learning Representations ICLR*, 2022.

22 Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3), 2001. doi:10.1145/502807.502810.

23 Martín Diéguez. Temporal answer set programming. In Agostino Dovier and Vítor Santos Costa, editors, *International Conference on Logic Programming, ICLP*, volume 17, pages 445–450, 2012. doi:10.4230/LIPICS.ICLP.2012.445.

24 Thomas Eiter and Mantas Simkus. Bidirectional answer set programs with function symbols. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 765–771, 2009. URL: http://ijcai.org/Proceedings/09/Papers/132.pdf.

25 Thomas Eiter and Mantas Simkus. FDNC: decidable nonmonotonic disjunctive logic programs with function symbols. *ACM Transactions on Computational Logic*, 11(2):14:1–14:50, 2010. doi:10.1145/1656242.1656249.

26 Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Ana Ozaki. On metric temporal description logics. In *European Conference on Artificial Intelligence ECAI*, volume 285, pages 837–845, 2016. doi:10.3233/978-1-61499-672-9-837.

27 Alex Ivliev, Lukas Gerlach, Simon Meusel, Jakob Steinberg, and Markus Krötzsch. Nemo: your friendly and versatile rule reasoning toolkit. In *International Conference on Principles of Knowledge Representation and Reasoning KR*, pages 743–754, 2024.

28 Elem Güzel Kalayci, Sebastian Brandt, Diego Calvanese, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. Ontology–based access to temporal data with Ontop: A framework proposal. *Int. J. Appl. Math.*, 29(1):17–30, 2019. doi:10.2478/AMCS-2019-0002.

29 Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Syst.*, 2(4):255–299, 1990. doi:10.1007/BF01995674.

30 Matthias Lanzinger, Markus Nissl, Emanuel Sallinger, and Przemysław Andrzej Wałęga. Temporal datalog with existential quantification. In *Proc. of IJCAI*, pages 3277–3285, 2023.

31 Marco Mori, Paolo Papotti, Luigi Bellomarini, and Oliver Giudice. Neural machine translation for fact-checking temporal claims. In *Extraction and VERification Workshop (FEVER)*, pages 78–82, 2022.

32 Boris Motik, Yavor Nenov, Robert Piro, Ian Horrocks, and Dan Olteanu. Parallel materialisation of Datalog programs in centralised, main-memory RDF systems. In *Conference on Artificial Intelligence AAAI*, pages 129–137, 2014. doi:10.1609/AAAI.V28I1.8730.

33 Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. Rdfox: A highly-scalable rdf store. In *International Semantic Web Conference*, pages 3–20. Springer, 2015. doi:10.1007/978-3-319-25010-6_1.

34 Markus Nissl and Emanuel Sallinger. Modelling smart contracts with DatalogMTL. In *Proc. of EDBT/ICDT*, 2022.

35 Katerina El Raheb, Theofilos Mailis, Vladislav Ryzhikov, Nicolas Papapetrou, and Yannis E. Ioannidis. Balonse: Temporal aspects of dance movement and its ontological representation. In *International Conference, ESWC*, volume 10250, pages 49–64, 2017. doi:10.1007/978-3-319-58451-5_4.

36 Vladislav Ryzhikov, Przemysław Andrzej Wałęga, and Michael Zakharyaschev. Data complexity and rewritability of ontology-mediated queries in metric temporal logic under the event-based semantics. In *Proc. of IJCAI*, pages 1851–1857, 2019.

37 Patrik Schneider, Daniel Alvarez-Coello, Anh Le-Tuan, Manh Nguyen-Duc, and Danh Le-Phuoc. Stream reasoning playground. In *The Semantic Web: 19th International Conference, ESWC*, pages 406–424, 2022. doi:10.1007/978-3-031-06981-9_24.

38 David J. Tena Cucala, Przemysław Andrzej Wałęga, Bernardo Cuenca Grau, and Egor V. Kostylev. Stratified negation in Datalog with metric temporal operators. In *Proc. of AAAI*, pages 6488–6495, 2021.

39 Veronika Thost. Metric temporal extensions of DL-Lite and interval-rigid names. In *Principles of Knowledge Representation and Reasoning KR*, pages 665–666. AAAI Press, 2018. URL: https://aaai.org/ocs/index.php/KR/KR18/paper/view/18035.

40 Dingmin Wang, Bernardo Cuenca Grau, Przemysław Andrzej Wałęga, and Pan Hu. Practical reasoning in datalogmtl. *Theory and Practice of Logic Programming*, 25(2):225–255, 2025. doi:10.1017/S1471068424000164.

41 Dingmin Wang, Pan Hu, Przemysław Andrzej Wałęga, and Bernardo Cuenca Grau. MeTeoR: Practical reasoning in Datalog with metric temporal operators. In *Proc. of AAAI*, pages 5906–5913, 2022.

42 Dingmin Wang, Przemysław Andrzej Wałęga, and Bernardo Cuenca Grau. Seminaive materialisation in DatalogMTL. In *International Joint Conference on Rules and Reasoning*, pages 183–197. Springer, 2022.

43 Dingmin Wang, Przemysław Andrzej Wałęga, and Bernardo Cuenca Grau. MTLearn: extracting temporal rules using datalog rule learners. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 21, pages 962–973, 2024.

44 Dingmin Wang, Bocheng Zou, and Zhen Han. tBen: Benchmarking and testing the rule-based temporal logic reasoning ability of large language models with datalogmtl, 2024.

45 Shaoyu Wang, Kaiyue Zhao, Dongliang Wei, Przemysław Andrzej Wałęga, Dingmin Wang, Hongming Cai, and Pan Hu. Goal-driven reasoning in datalogmtl with magic sets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 15203–15211, 2025.

46 Przemysław Andrzej Wałęga, Bernardo Cuenca Grau, Mark Kaminski, and Egor V. Kostylev. DatalogMTL: Computational complexity and expressive power. In *Proc. of IJCAI*, pages 1886–1892, 2019.

47 Przemysław Andrzej Wałęga, Bernardo Cuenca Grau, Mark Kaminski, and Egor V. Kostylev. DatalogMTL over the integer timeline. In *Proc. of KR*, pages 768–777, 2020.

48 Przemysław Andrzej Wałęga, Bernardo Cuenca Grau, Mark Kaminski, and Egor V. Kostylev. Tractable fragments of Datalog with metric temporal operators. In *Proc. of IJCAI*, pages 1919–1925, 2020.

49 Przemysław Andrzej Wałęga, Mark Kaminski, and Bernardo Cuenca Grau. Reasoning over streaming data in metric temporal Datalog. In *Proc. of AAAI*, pages 3092–3099, 2019.

50 Przemysław Andrzej Wałęga, Mark Kaminski, Dingmin Wang, and Bernardo Cuenca Grau. Stream reasoning with DatalogMTL. *J. Web Semant.*, 76:100776, 2023. doi:10.1016/J.WEBSEM.2023.100776.

51 Przemysław Andrzej Wałęga and Michael Rawson. Expressive power of temporal message passing. In *Proc. of AAAI*, pages 21000–21008, 2025.

52 Przemysław Andrzej Wałęga, David J. Tena Cucala, Egor V. Kostylev, and Bernardo Cuenca Grau. DatalogMTL with negation under stable models semantics. In *Proc. of KR*, pages 609–618, 2021.

53 Przemysław Andrzej Wałęga, Michał Zawidzki, and Bernardo Cuenca Grau. Finitely materialisable Datalog programs with metric temporal operators. In *Proc. of KR*, pages 619–628, 2021.

54 Przemysław Andrzej Wałęga, Michał Zawidzki, and Bernardo Cuenca Grau. Finite materialisability of Datalog programs with metric temporal operators. *J. Artif. Intell. Res.*, 76, 2023. doi:10.1613/JAIR.1.14040.

55 Przemysław Andrzej Wałęga, Michał Zawidzki, Dingmin Wang, and Bernardo Cuenca Grau. Materialisation-based reasoning in DatalogMTL with bounded intervals. In *AAAI Conference on Artificial Intelligence*, pages 6566–6574, 2023.