Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning
KR in the Wild

Efficient Temporal Reasoning with Non-Temporal Engines: Embedding DatalogMTL into Datalog

Mathijs van Noort[1] , **Przemysław Andrzej Wałęga**[2]
[1]IDLab, Ghent University-imec, BE
[2]School of Electronic Engineering and Computer Science, Queen Mary University of London, UK
mathijs.vannoort@ugent.be, p.walega@qmul.ac.uk

Abstract

We present a translation from DatalogMTL, a temporal extension of Datalog rule language with operators from metric temporal logic, into non-temporal Datalog with arithmetics. As we prove, the translation preserves key semantic properties such as entailment, consistency, and finiteness of materialisability. As a result, we obtain a faithful embedding of DatalogMTL into classical Datalog, enabling complex temporal reasoning tasks to be executed without the need for specialised temporal reasoning engines. We implement and evaluate this translation using three state-of-the-art Datalog systems: Nemo, EYE, and Eyelet. Remarkably, non-temporal Datalog engines are able to achieve comperable performance, and in some cases outperform dedicated temporal reasoning engines, demonstrating the practical viability and efficiency of the translation.

1 Introduction

Temporal reasoning is central in modern data-intensive systems, where information evolves continuously and analytical systems must detect complex temporal relationships. Domains such as financial monitoring, cybersecurity, transportation analysis, and IoT ecosystems frequently require identifying temporally extended patterns, e.g. recurring behaviours, suspicious event sequences, or causal chains unfolding over time. In many of these settings, temporal constraints must be articulated explicitly and in an interpretable manner, making rule-based temporal languages an attractive option. Yet, building temporal reasoning engines remains a demanding engineering challenge, especially in the presence of dense time, data stamped with intervals, and non-trivial temporal modalities, while maintaining correct semantics and practical efficiency. Temporal engines tend to be significantly less mature than widely deployed non-temporal systems. This mismatch between the practical demand for temporal reasoning and the limited maturity of temporal systems motivates our exploration of methods that allow for temporal reasoning by leveraging existing and well-optimised non-temporal systems.

In this paper we focus on DatalogMTL (Brandt et al. 2018)—an extension of the state-of-the-art rule-based language Datalog (Abiteboul, Hull, and Vianu 1995) with metric temporal logic (MTL) operators. It 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 (Brandt et al. 2017), economic fact-checking (Mori et al. 2022), human movement description (Raheb et al. 2017), and verification of banking agreements (Nissl and Sallinger 2022a).

To illustrate capabilities of DatalogMTL, consider a simplified banking system that monitors transactions to detect suspicious behaviour. The system logs transactions of the form $Trans(acct1054, acct0321)@11023$, indicating that at timestamp 11023 a transfer was recorded from account $acct1054$ to account $acct0321$. To prevent money laundering, the bank aims to detect transaction loops, where money is sent from one account to another and then returned shortly after. This can be expressed in DatalogMTL using the following rules:

$$\boxplus_{[0,14]} Susp(x) \leftarrow Trans(y,x) \wedge \diamondminus_{[0,5]} Trans(x,y), \quad (1)$$

$$RedList(x) \leftarrow \boxminus_{[0,30]} Susp(x). \quad (2)$$

Rule (1) flags an account x as suspicious continuously for the next interval of 14 days (expressed with $\boxplus_{[0,14]} Susp(x)$) when x receives a transaction from another account y ($Trans(y,x)$) and x had previously sent money to y sometime within the past 5 days ($\diamondminus_{[0,5]} Trans(x,y)$). Rule (2), in turn, places x on the red list ($RedList(x)$) if x has been continuously flagged as suspicious for the past 30 days ($\boxminus_{[0,30]} Susp(x)$).

DatalogMTL offers a good balance between the expressive power and computational complexity of reasoning (Wałęga et al. 2019; Wałęga et al. 2021; Bellomarini, Nissl, and Sallinger 2021b), making it a versatile tool for a wide range of temporal reasoning applications. Consequently, research on DatalogMTL has increasingly focused on developing practical reasoning algorithms. This has led to several systems employing diverse reasoning techniques, including ontology-based rewritings (Kalaycı et al. 2018), automata-theoretic constructions (Wang et al. 2022), materialisation-based approaches (Bellomarini et al. 2022; Bellomarini, Sallinger, and Gottlob 2018), saturation procedures (Wałęga et al. 2023), semi-naïve evaluations (Wang et al. 2025a), and magic set rewritings (Wang et al. 2025b).

However, all these approaches require sophisticated mechanisms tailored to the complex setting of DatalogMTL. Developing such systems is challenging, prone to errors, and

the resulting tools often suffer from limited maintainability. This stands in contrast to standard (non-temporal) Datalog systems, which are mature, robust, and used in industry. They are maintained and optimised by professional developers, and offered as reliable commercial products (Nenov et al. 2015; Bellomarini, Sallinger, and Gottlob 2018).

This leads us to the following research questions:

- Can the complex task of temporal reasoning in DatalogMTL be effectively performed using existing, non-temporal Datalog engines?
- If so, which theoretical properties are preserved by such an approach, and how does its practical performance compare to that of dedicated DatalogMTL reasoners?

We address these questions and show a translation to Datalog that provides us with a practical tool for reasoning in DatalogMTL. Our main contributions are as follows:

- In Section 4 we construct a translation of a pair of a DatalogMTL program Π and dataset $\mathcal{D}$ into a Datalog (with arithmetic operators) program $\mathbf{tr}(\Pi)$ and a set of Datalog facts $\mathbf{tr}(\mathcal{D})$.
- We also establish a correspondence between materialisation steps in DatalogMTL and in the translated Datalog setting. Materialisation terminates for Π and $\mathcal{D}$ if and only if it terminates for $\mathbf{tr}(\Pi)$ and $\mathbf{tr}(\mathcal{D})$. Moreover, consistency of Π and $\mathcal{D}$ implies consistency of $\mathbf{tr}(\Pi)$ and $\mathbf{tr}(\mathcal{D})$, whereas the converse holds whenever Π and $\mathcal{D}$ are finitely materialisable.
- In Section 5 we turn to practical aspects of the translation. We show how it can be adapted to the N3 triple-store format used by engines such as EYE.
- Then, we provide implementations of the translations based on Datalog and on the N3 triple formats. We compare the performance of the state-of-the-art DatalogMTL reasoner MeTeoR with our translated programs executed in the Datalog engines Nemo and Eyelet as well as in the N3 reasoner EYE. As we show, in several cases Nemo outperforms MeTeoR.

Our results indicate that, instead of developing dedicated DatalogMTL reasoners from scratch, a promising alternative is to translate DatalogMTL programs into the standard Datalog setting and exploit existing, optimised reasoning engines. This approach not only simplifies implementation but also enables seamless integration of temporal and non-temporal reasoning within a single reasoning framework.

2 Related Work

The extension of traditional reasoning with temporal dimension has been a topic of continued research. Following early work on static knowledge representation and reasoning, several frameworks have incorporated time into established rule-based or logic-based paradigms. Examples include temporal extensions of Answer Set Programming (ASP), such as C-ASP (Pham, Ali, and Mileo 2019) and PrASP (Nickles and Mileo 2014). Within the Semantic Web community, significant efforts have focused on extending SPARQL to support streaming data, leading to formalisms such as RSP-QL (Dell'Aglio et al. 2014), C-SPARQL (Barbieri et al. 2010), CQELS (Phuoc 2013), and EP-SPARQL (Anicic et al. 2012). These systems typically rely on windowing mechanisms: temporal data is divided into bounded subsets that can then be processed by non-temporal engines. For example, DIVIDE (De Brouwer et al. 2023) relies on internal RDF reasoners like RDFox (Nenov et al. 2015), EYE (De Meester et al. 2015) uses Prolog reasoner, and some others employ CEL DL reasoner (Lécué 2012).

A well-known limitation of window-based stream reasoning is that discarding the full temporal dimension prevents these systems from recognising extended temporal relations or long-range dependencies, constraining their ability to detect and analyse complex patterns, often referred to as Complex Event Processing (CEP). This reflects a broader trade-off: such approaches benefit from established non-temporal engines but lose expressive power for temporal pattern recognition. Another challenge in the stream reasoning landscape is the absence of a unifying semantic framework. Existing formalisms differ substantially in how they interpret windows, timestamps, incremental updates, or the interaction between temporal and logical operators. Some adopt windows as preprocessing steps that generate static sub-datasets, while others integrate temporal modalities directly into rule languages, yielding different expressive capabilities and evaluation behaviours. As a consequence, these systems often lack interoperability, complicate comparisons across formalisms, and make it difficult to establish common benchmarks or correctness guarantees.

An alternative line of work integrates temporal reasoning natively into rule-based systems. Examples include LARS (Beck, Dao-Tran, and Eiter 2018), DatalogMTL, and RTEC (Artikis, Sergot, and Paliouras 2012), which incorporate additional temporal operators. These approaches require dedicated implementations, such as MeTeoR (Wang et al. 2022), Ticker (Beck 2018), and Lazer (Bazoobandi, Beck, and Urbani 2017). Temporal reasoning has also been explored in temporal description logics (Gutiérrez-Basulto, Jung, and Ozaki 2016; Baader et al. 2017; Thost 2018; Artale and Franconi 1998) and other temporal variants of Datalog (Chomicki and Imielinski 1988; Artale et al. 2015; Kontchakov et al. 2016).

The line of research on DatalogMTL has provided an in-depth theoretical analysis. Its computational complexity has been established for both data and combined complexity (Brandt et al. 2018; Wałęga et al. 2019; Wałęga et al. 2021). Alternative semantics have been developed, including event-based semantics (Ryzhikov, Wałęga, and Zakharyaschev 2019) and integer-based timelines (Wałęga et al. 2020; Wałęga, Zawidzki, and Cuenca Grau 2021). Further extensions introduce non-monotonic negation (Cucala et al. 2021; Wałęga et al. 2021), existential rules (Lanzinger et al. 2023), and temporal aggregation (Bellomarini, Nissl, and Sallinger 2021a). Finite materialisability, including developing algorithms for its checking and establishing computational complexity of this problem, have been studied for DatalogMTL (Wałęga, Zawidzki, and Cuenca Grau 2023;

Wałęga, Zawidzki, and Cuenca Grau 2021). DatalogMTL has also been studied in the context of inconsistency handling (Bienvenu, Bourgaux, and Khodadaditaghanaki 2025) and temporal rule mining (Wang, Wałęga, and Cuenca Grau 2024). Applications span smart contracts (Colombo et al. 2023; Nissl and Sallinger 2022a; Nissl and Sallinger 2022b), crypto-activity analysis (Colombo et al. 2024), economic fact-checking (Mori et al. 2022), and human movement description (Raheb et al. 2017).

Several practical reasoning approaches for DatalogMTL have emerged, including extensions of existing platforms such as Ontop (Kalaycı et al. 2018) and Vadalog (Bellomarini et al. 2022). Standalone implementations include systems based on automata (Wang et al. 2022), saturation procedures (Wałęga et al. 2023), semi-naïve evaluation (Wang et al. 2025a), and magic-set rewriting (Wang et al. 2025b). A dedicated generator, iTemporal (Bellomarini, Nissl, and Sallinger 2022), provides benchmarks and synthetic datasets for temporal Datalog reasoning.

3 Background

We will introduce syntax and semantics of $\text{Datalog}_\mathbb{Q}$, that is, Datalog with arithmetics over rationals, as well as of DatalogMTL, that is, Datalog with metric temporal operators.

$\text{Datalog}_\mathbb{Q}$. For $\text{Datalog}_\mathbb{Q}$ we assume a two-sorted language, where each constant and variable is either of an *object* or a *numeric* sort. *Object terms* are object constants and variables, whereas *numeric terms* are defined inductively as numeric constants (values in $\mathbb{Q} \cup \{-\infty, +\infty\}$), variables, and arithmetic expressions $|t|, t+t', t-t', t\times t', t/t'$, for any numeric terms t and t'. We will also use $\max(t_1, \dots, t_k)$ and $\min(t_1, \dots, t_k)$, which are definable using the other functions, for example, $\max(t, t') = (t+t'+|t-t'|)/2$. *Relational atoms* are of the form $P(\mathbf{s})$, for P a predicate and $\mathbf{s}$ a tuple of terms with arity matching P. *Comparison atoms* are of the forms $(t = t')$ and $(t \leq t')$, for t and t' being numeric terms. A $\text{Datalog}_\mathbb{Q}$ *rule* is of the form $A' \leftarrow A_1 \wedge \ldots \wedge A_n$, where the *rule head* A', and each *rule body atom* A_i is a (relational or comparison) atom, $\top$, or $\bot$. The conjunction $A_1 \wedge \cdots \wedge A_n$ is called the *rule body*. A $\text{Datalog}_\mathbb{Q}$ *program*, Λ, is a finite set of $\text{Datalog}_\mathbb{Q}$ rules. An expression (term, atom, rule, etc.) is *ground* if it does not mention variables.

A $\text{Datalog}_\mathbb{Q}$ *interpretation*, I, is a (possibly infinite) set of ground relational atoms. In each interpretation, we let $I \models \top$ and $I \not\models \bot$. For a ground relational atom A we let $I \models A$ if $A \in I$ after the standard evaluation of arithmetic operations in A. For a ground comparison atom, we let $I \models (t = t')$ or $I \models (t \leq t')$, if after evaluation of arithmetic operations, the (in-)equality holds over the rational numbers. An interpretation I is a *model* of a rule $A' \leftarrow A_1 \wedge \cdots \wedge A_n$, if for every *grounding* ν (a mapping of variables to constants of the same sort), $I \models \nu(A_i)$ for all $i \in \{1, \dots, n\}$ implies that $I \models \nu(A')$. Interpretation I is a *model* of a program Λ if it is a model of all rules in Λ, and Λ is *consistent* if it has a model. Furthermore, Λ *entails* a ground atom A, in symbols $\Lambda \models A$, if $I \models A$ in every model I of Λ.

A procedural counterpart of the above semantics is defined by means of the *immediate consequence operator* T_Λ, for a program Λ. We let T_Λ be a partial function which maps an interpretation I to the unique subset-minimal interpretation $T_\Lambda(I)$ if one exists (otherwise it is undefined), such that $I \subseteq T_\Lambda(I)$ and for any rule $A' \leftarrow A_1 \wedge \cdots \wedge A_n$ in Λ and any grounding ν, the fact that $I \models \nu(A_i)$ for all $i \in \{1, \dots, n\}$ implies that $T_\Lambda(I) \models \nu(A')$. Note that $T_\Lambda(I)$ is undefined precisely when Λ contains a rule with $\bot$ in the head and with body satisfied in I. We let $T_\Lambda^{k+1}(I) = T_\Lambda\big(T_\Lambda^k(I)\big)$, for $k \in \mathbb{N}$ and $T_\Lambda^\omega(I) = \bigcup_{k\in\mathbb{N}} T_\Lambda^k(I)$. The *canonical interpretation*, $\mathfrak{C}_\Lambda$, is defined as $T_\Lambda^\omega(\emptyset)$. It can be shown that Λ is consistent if $\mathfrak{C}_\Lambda$ is well-defined; moreover Λ entails A, if Λ is inconsistent or $\mathfrak{C}_\Lambda \models A$. We say that Λ is *finitely materialisable* if there is $k \in \mathbb{N}$ for which $T_\Lambda^k(\emptyset) = \mathfrak{C}_\Lambda$. Similarly, a pair consisting of a program Λ and an interpretation I is consistent if $T_\Lambda^\omega(I)$ is well-defined, and the pair is finitely materialisable if $\mathfrak{C}_\Lambda(I) \coloneqq T_\Lambda^\omega(I) = T_\Lambda^k(I)$, for some $k \in \mathbb{N}$.

DatalogMTL. In contrast to $\text{Datalog}_\mathbb{Q}$, the language of DatalogMTL has one (object) sort. Terms are constants and variables, whereas relational atoms are defined as in $\text{Datalog}_\mathbb{Q}$. *Metric atoms* are defined recursively as $\top$, $\bot$, relational atoms, $\boxplus_\varrho M$, $\boxminus_\varrho M$, $\diamondplus_\varrho M$, $\diamondminus_\varrho M$, $M \mathcal{S}_\varrho M$, and $M \mathcal{U}_\varrho M$, where M are metric atoms and ϱ is a closed, non-empty, and positive (i.e. including only positive numbers) interval over the rational timeline; it can be right-unbounded, i.e. with $+\infty$ as the right endpoint. A *DatalogMTL rule* is of the form $M' \leftarrow M_1 \wedge \ldots \wedge M_n$, where each M_i is a metric atom and M' is a metric atom not mentioning any of $\diamondplus$, $\diamondminus$, $\mathcal{S}$, and $\mathcal{U}$. A *DatalogMTL program* is a finite set of DatalogMTL rules. A DatalogMTL program is *flat* if it does not mention any nesting of temporal operators. Unlike in $\text{Datalog}_\mathbb{Q}$, rules of DatalogMTL do not allow us to express facts. A DatalogMTL *fact* is of the form $P(\mathbf{c})@\varrho$, where $P(\mathbf{c})$ is a ground relational atom and ϱ is a closed interval over rationals; we allow for left- and right-unbounded intervals (with $-\infty$ and $+\infty$ as endpoints). In case of *punctual* intervals, $[t, t]$, we may write $P(\mathbf{c})@t$ instead of $P(\mathbf{c})@[t, t]$. A *DatalogMTL dataset* is a finite set of temporal facts. A *DatalogMTL interpretation* $\mathfrak{I}$ is a function mapping each *time point* (a rational number) to a set of ground relational atoms. We have $\mathfrak{I}, t \models \top$ and $\mathfrak{I}, t \not\models \bot$, for every t. We let $\mathfrak{I}, t \models P(\mathbf{c})$ if $P(\mathbf{c}) \in \mathfrak{I}(t)$. Then, for every ground metric atom M we have:

$$
\begin{aligned}
\mathfrak{I}, t \models \diamondminus_\varrho M \quad & \text{iff } \mathfrak{I}, t' \models M \text{ for some } t' \text{ with } t - t' \in \varrho,\\
\mathfrak{I}, t \models \diamondplus_\varrho M \quad & \text{iff } \mathfrak{I}, t' \models M \text{ for some } t' \text{ with } t' - t \in \varrho,\\
\mathfrak{I}, t \models \boxminus_\varrho M \quad & \text{iff } \mathfrak{I}, t' \models M \text{ for all } t' \text{ with } t - t' \in \varrho,\\
\mathfrak{I}, t \models \boxplus_\varrho M \quad & \text{iff } \mathfrak{I}, t' \models M \text{ for all } t' \text{ with } t' - t \in \varrho,\\
\mathfrak{I}, t \models M_1 \mathcal{S}_\varrho M_2 \quad & \text{iff } \mathfrak{I}, t' \models M_2 \text{ for some } t' \text{ with } t - t' \in \varrho,\\
& \qquad \text{and } \mathfrak{I}, t'' \models M_1 \text{ for all } t'' \in (t', t),\\
\mathfrak{I}, t \models M_1 \mathcal{U}_\varrho M_2 \quad & \text{iff } \mathfrak{I}, t' \models M_2 \text{ for some } t' \text{ with } t' - t \in \varrho,\\
& \qquad \text{and } \mathfrak{I}, t'' \models M_1 \text{ for all } t'' \in (t, t').
\end{aligned}
$$

We write $\mathfrak{I} \models M@\varrho$ if $\mathfrak{I}, t \models M$ for all $t \in \varrho$.

An interpretation $\mathfrak{I}$ is a *model* of a rule $M' \leftarrow M_1 \wedge \ldots \wedge M_n$ if for every time point t and for every grounding ν, it holds that $\mathfrak{I}, t \models \nu(M')$, whenever $\mathfrak{I}, t \models \nu(M_i)$ for every $i \in \{1, \ldots, n\}$. Interpretation $\mathfrak{I}$ is a model of a program Π if it is a *model* of every rule in Π. An interpretation is a *model* of a dataset $\mathcal{D}$ if for every fact $P(\mathbf{c})@\varrho \in \mathcal{D}$, we have $\mathfrak{I} \models P(\mathbf{c})@\varrho$. We let $\mathfrak{I}_\mathcal{D}$ be the unique interpretation which is the minimal model of $\mathcal{D}$. A pair $(\Pi, \mathcal{D})$ of a program and dataset is *consistent* if there is an interpretation that is a model for both Π and $\mathcal{D}$. A pair $(\Pi, \mathcal{D})$ *entails* a ground metric atom M at t, written as $(\Pi, \mathcal{D}), t \models M$, if $\mathfrak{I}, t \models M$ for every model $\mathfrak{I}$ of $(\Pi, \mathcal{D})$.

Similarly to Datalog$_\mathbb{Q}$, an alternative semantics of DatalogMTL is provided using the *immediate consequence operator* T_Π, which is a partial function mapping an interpretation $\mathfrak{I}$ to the unique interpretation $T_\Pi(\mathfrak{I})$, if one exists (otherwise it is undefined), such that for each t, $T_\Pi(\mathfrak{I})(t)$ is the subset-minimal set satisfying: (i) $\mathfrak{I}(t) \subseteq T_\Pi(\mathfrak{I})(t)$ and (ii) for every rule $M' \leftarrow M_1 \wedge \ldots \wedge M_n$ in Π and any grounding ν, we have $\mathfrak{I}, t \models \nu(M')$ whenever $\mathfrak{I}, t \models \nu(M_i)$ for every $i \in \{1, \ldots, n\}$. For any successor ordinal α, we let $T_\Pi^{\alpha+1}(\mathfrak{I}) = T_\Pi(T_\Pi^\alpha(\mathfrak{I}))$, and for a limit ordinal γ, we let $T_\Pi^\gamma(\mathfrak{I}) = \bigcup_{\beta<\gamma} T_\Pi^\beta(\mathfrak{I})$. The canonical interpretation for program Π and dataset $\mathcal{D}$ is $\mathfrak{C}_{\Pi,\mathcal{D}} = T_\Pi^{\omega_1}(\mathfrak{I}_\mathcal{D})$, where ω_1 is the first uncountable ordinal. Then $(\Pi, \mathcal{D})$ is *consistent* if $\mathfrak{C}_{\Pi,\mathcal{D}}$ is well-defined, whereas $(\Pi, \mathcal{D})$ *entails* a fact $P(\mathbf{c})@\varrho$, in symbols $(\Pi, \mathcal{D}) \models P(\mathbf{c})@\varrho$, if $(\Pi, \mathcal{D})$ is inconsistent or $\mathfrak{C}_{\Pi,\mathcal{D}} \models P(\mathbf{c})@\varrho$. Moreover, $(\Pi, \mathcal{D})$ is *finitely materialisable* if there is $k \in \mathbb{N}$ for which $T_\Pi^k(\mathfrak{I}_\mathcal{D}) = \mathfrak{C}_{\Pi,\mathcal{D}}$. The canonical interpretation $\mathfrak{C}_{\Pi,\mathcal{D}}$ of a consistent pair $(\Pi, \mathcal{D})$ is the minimal model of this pair (Brandt et al. 2017).

4 Translating DatalogMTL to Datalog$_\mathbb{Q}$

In this section, we will show how to translate DatalogMTL into Datalog$_\mathbb{Q}$, to simulate temporal reasoning with non-temporal Datalog engines. For readability we assume that DatalogMTL programs are flat, i.e. with no nested temporal operators. Any program can be translated into this form (Brandt et al. 2018). Moreover, until Proposition 10, we assume that programs do not mention $\bot$ in rule heads. In this setting no inconsistency can occur, and $T_\Pi^\gamma(\mathfrak{I})$ is well-defined for any γ. In Proposition 10 we turn our attention to programs with $\bot$, as we will consider consistency checking.

We start by providing intuition regarding a desired relation between DatalogMTL and its translation into Datalog. To this end, consider again a DatalogMTL program Π consisting of Rules (1) and (2), which detect circular financial transactions. Assume that the DatalogMTL dataset $\mathcal{D}$ gathered by the bank contains facts $Trans(adam, betty)@t$ and $Trans(betty, adam)@t+2$, for time points $t \in \{0, 10, 20\}$. Then, by application of Rule (1), $Susp(adam)$ is derived in intervals $[2, 16]$, $[12, 26]$, and $[22, 36]$, as depicted in Figure 1. Hence, we obtain $Susp(adam)@[2, 36]$, and so, an application of Rule (2) derives $RedList(adam)@[32, 36]$.

The aim is to translate Π into a Datalog$_\mathbb{Q}$ program $\mathbf{tr}(\Pi)$ and $\mathcal{D}$ into a Datalog$_\mathbb{Q}$ interpretation $\mathbf{tr}(\mathcal{D})$ such that facts derived by $\mathbf{tr}(\Pi)$ and $\mathbf{tr}(\mathcal{D})$ can correspond to the temporal facts derived by Π and $\mathcal{D}$. For example, we aim to simulate the derivation of $Susp(adam)[2, 36]$ in DatalogMTL with a derivation of $Susp(adam, 2, 36)$ in Datalog$_\mathbb{Q}$. In other words, we want to encode in Datalog$_\mathbb{Q}$ the endpoints of intervals in which a fact $Susp(adam)$ holds in DatalogMTL, by adding two numerical positions in predicates.

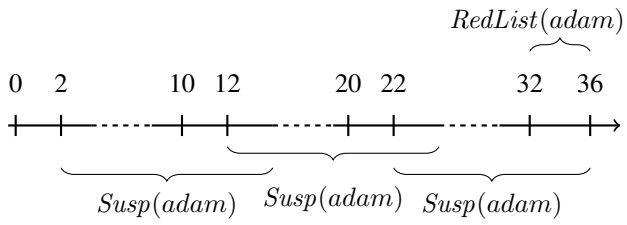


Figure 1: Facts derived by Rules (1) and (2)

It is worth noting, however, that due to the continuous timeline and the need to consider coalescing of intervals, the correspondence between DatalogMTL and Datalog$_\mathbb{Q}$ is not straightforward. For example, it may be tempting to assume that whenever $P(\mathbf{c})$ is entailed at some interval $[a, b]$ in DatalogMTL, then $P(\mathbf{c}, a, b)$ should be derived in Datalog$_\mathbb{Q}$. Notice that if, for instance, $P(\mathbf{c})$ is entailed in $[0, 1]$, this would mean that $P(\mathbf{c}, a, b)$ needs to be derived for all a, b such that $[a, b] \subseteq [0, 1]$. Since we consider a dense (rational) time line, there are infinitely many such intervals, so simulating a single DatalogMTL fact $P(\mathbf{c})@[0, 1]$ would require an infinite Datalog$_\mathbb{Q}$ interpretation, which is not feasible in practice. An alternative option could be to require that $P(\mathbf{c}, a, b)$ is entailed in Datalog only for the maximal interval $[a, b]$ such that $P(\mathbf{c})@[a, b]$ is entailed in DatalogMTL. This would not require infinite Datalog$_\mathbb{Q}$ interpretations. However, it would be too restrictive, as it would not allow for coalescing intervals. Indeed, as presented in the example from Figure 1, it is essential to first derive $Susp(adam)@[2, 16]$, $Susp(adam)@[12, 26]$, and $Susp(adam)@[22, 36]$, then coalesce these intervals to obtain $Susp(adam)@[2, 36]$, and only then apply Rule (2). We would not be able to simulate such derivations in Datalog$_\mathbb{Q}$ if we did not allow for storing facts over intervals which have non-empty intersections.

This leads us to a more complicated requirement defined next. Intuitively we will consider a DatalogMTL interpretation $\mathfrak{I}$ as corresponding to a Datalog$_\mathbb{Q}$ interpretation I if $\mathfrak{I}$ pointwise subsumes I, and vice versa. By pointwise subsumption we mean that if $\mathfrak{I}, t \models P(\mathbf{c})$ in DatalogMTL, then $P(\mathbf{c}, a, b) \in I$ for some $t \in [a, b]$ in Datalog$_\mathbb{Q}$. For the opposite subsumption, we require that $P(\mathbf{c}, a, b) \in I$ in Datalog$_\mathbb{Q}$ implies $\mathfrak{I}, t \models P(\mathbf{c})$ for each $t \in [a, b]$ in DatalogMTL. We formalise these notions in the definition below.

Definition 1. A DatalogMTL interpretation $\mathfrak{I}$ is *pointwise subsumed* by a Datalog$_\mathbb{Q}$ interpretation I, denoted $\mathfrak{I} \leq_\mathrm{p} I$, if for any relational atom $P(\mathbf{c})$ and time point t with $\mathfrak{I}, t \models P(\mathbf{c})$, there exists an interval $[a, b]$ with $P(\mathbf{c}, a, b) \in I$ and $t \in [a, b]$. Conversely, we say that I is *pointwise subsumed* by $\mathfrak{I}$, $I \leq_\mathrm{p} \mathfrak{I}$, if for every time point t and interval $[a, b]$ with

$t \in [a,b]$ and $P(\mathbf{c},a,b) \in I$, it holds that $\mathfrak{I}, t \models P(\mathbf{c})$. If $\mathfrak{I} \leq_{\text{p}} I$ and $I \leq_{\text{p}} \mathfrak{I}$, we write $\mathfrak{I} =_{\text{p}} I$.

In the rest of this section we show how to construct a translation of DatalogMTL programs and datasets into $\text{Datalog}_{\mathbb{Q}}$ so that we obtain a correspondence between DatalogMTL and $\text{Datalog}_{\mathbb{Q}}$ interpretations in terms of the relation $=_{\text{p}}$.

The translation of DatalogMTL facts is straightforward, namely we will translate each fact $P(\mathbf{c})@[a,b]$ into $P(\mathbf{c},a,b)$. For example, $\mathit{Trans}(\mathit{adam},\mathit{betty})@0$ is translated into $\mathit{Trans}(\mathit{adam},\mathit{betty},0,0)$, since a punctual interval 0 starts and ends in the same time point 0. The translation of DatalogMTL rules is much more complicated and is obtained in two steps. The first step is to translate temporal operators in the rule into arithmetic operations. For example, Rule (2) is translated into the following $\text{Datalog}_{\mathbb{Q}}$ rule:

$$\begin{aligned}&\mathit{RedList}(x,t^-,t^+) \leftarrow \mathit{Susp}(x,t_1^-,t_1^+) \wedge \\ &(t^- = t_1^- + 30) \wedge (t^+ = t_1^+ + 0) \wedge (t^- \leq t^+)\end{aligned}$$

This rule states that $\mathit{Susp}(x,t_1^-,t_1^+)$ allows us to derive $\mathit{Susp}(x,t_1^-+30,t_1^+)$ if $t_1^-+30 \leq t_1^+$. Note that the last condition guarantees that we derive facts only over intervals, that is, the first endpoint cannot be greater than the second.

The second step of the translation is to add $\text{Datalog}_{\mathbb{Q}}$ rules for coalescing intervals. For example to coalesce facts over the predicate Susp, we use the rule

$$\begin{aligned}\mathit{Susp}(x,t_1,t_4) \leftarrow &\mathit{Susp}(x,t_1,t_2) \wedge \mathit{Susp}(x,t_3,t_4) \wedge \\ &(t_3 \leq t_2) \wedge (t_1 \leq t_4),\end{aligned}$$

which states that from $\mathit{Susp}(x,a,b)$ and $\mathit{Susp}(x,c,d)$ over intersecting intervals $[a,b]$ and $[c,d]$, we can derive $\mathit{Susp}(x,t_1,t_4)$ over the union $[t_1,t_4] = [a,b] \cup [c,d]$.

The full translation requires considering all the possible temporal operators and handling conjunctions in rule bodies. The formal definition is provided bellow.

Definition 2. For a DatalogMTL dataset $\mathcal{D}$ we define

$$\mathbf{tr}(\mathcal{D}) \coloneqq \{P(\mathbf{s},a,b) \mid P(\mathbf{s})@[a,b] \in \mathcal{D}\}.$$

For a DatalogMTL rule $r = M' \leftarrow M_1 \wedge \ldots \wedge M_n$, we let

$$\begin{aligned}\mathbf{tr}(r) \coloneqq A' \leftarrow A_1 \wedge \cdots \wedge A_n \wedge (t^- = \max_{i \leq n} \varrho_i^-) \wedge \\ (t^+ = \min_{i \leq n} \varrho_i^+) \wedge (t^- \leq t^+),\end{aligned} \tag{3}$$

where A' is as follows:

- $A' \coloneqq M'$, if M' is $\top$ or $\bot$,
- $A' \coloneqq P(\mathbf{s},t^-,t^+)$, if M' is $P(\mathbf{s})$,
- $A' \coloneqq P(\mathbf{s},t^-+a,t^++b)$, if M' is $\boxplus_{[a,b]}P(\mathbf{s})$,
- $A' \coloneqq P(\mathbf{s},t^--b,t^+-a)$, if M' is $\boxminus_{[a,b]}P(\mathbf{s})$,

expressions A_i are as follows:

- $A_i \coloneqq M_i$, if M_i is $\top$ or $\bot$,
- $A_i \coloneqq P(\mathbf{s},t_i^-,t_i^+)$, if M_i is $P(\mathbf{s})$, $\diamondplus_{[a,b]}P(\mathbf{s})$, or $\diamondminus_{[a,b]}P(\mathbf{s})$,
- $A_i \coloneqq P(\mathbf{s},t_i^-,t_i^+) \wedge (b-a) \leq (t_i^+ - t_i^-)$, if M_i is $\boxplus_{[a,b]}P(\mathbf{s})$ or $\boxminus_{[a,b]}P(\mathbf{s})$,
- $A_i \coloneqq P_1(\mathbf{s},t_{i1}^-,t_{i1}^+) \wedge P_2(\mathbf{s},t_{i2}^-,t_{i2}^+) \wedge (t_{i1}^- \leq t_{i2}^- - b) \wedge (t_{i2}^+ \leq t_{i1}^+)$, if M_i is $P_1(\mathbf{s})\mathcal{U}_{[a,b]}P_2(\mathbf{s})$,
- $A_i \coloneqq P_1(\mathbf{s},t_{i1}^-,t_{i1}^+) \wedge P_2(\mathbf{s},t_{i2}^-,t_{i2}^+) \wedge (t_{i1}^- \leq t_{i2}^-) \wedge (t_{i2}^+ + b \leq t_{i1}^+)$, if M_i is $P_1(\mathbf{s})\mathcal{S}_{[a,b]}P_2(\mathbf{s})$,

and intervals $\varrho_i \coloneqq [\varrho_i^-,\varrho_i^+]$ are defined as

- $\varrho_i \coloneqq [-\infty,+\infty]$ for $M_i = \top$ or $= \bot$,
- $\varrho_i \coloneqq [t_i^-,t_i^+]$ if M_i is $P(\mathbf{s})$,
- $\varrho_i \coloneqq [t_i^- - b, t_i^+ - a]$ if M_i is $\diamondplus_{[a,b]}P(\mathbf{s})$,
- $\varrho_i \coloneqq [t_i^- + a, t_i^+ + b]$ if M_i is $\diamondminus_{[a,b]}P(\mathbf{s})$,
- $\varrho_i \coloneqq [t_i^- - a, t_i^+ - b]$ if M_i is $\boxplus_{[a,b]}P(\mathbf{s})$,
- $\varrho_i \coloneqq [t_i^- + b, t_i^+ + a]$ if M_i is $\boxminus_{[a,b]}P(\mathbf{s})$,
- $\varrho_i \coloneqq [t_{i2}^- - b, t_{i2}^+ - a]$ if M_i is $P_1(\mathbf{c}_1)\mathcal{U}_{[a,b]}P_2(\mathbf{s}_2)$,
- $\varrho_i \coloneqq [t_{i2}^- + a, t_{i2}^+ + b]$ if M_i is $P_1(\mathbf{c}_1)\mathcal{S}_{[a,b]}P_2(\mathbf{s}_2)$,

Finally, for a DatalogMTL program Π we define

$$\mathbf{tr}(\Pi) \coloneqq \Pi_{\mathit{time}} \cup \bigcup_{r \in \Pi} \mathbf{tr}(r),$$

where Π_{time} is the program consisting of the following rule, for every predicate P in Π, where $\mathbf{x}$ matches the arity of P:

$$\begin{aligned}P(\mathbf{x},t_1,t_4) \leftarrow &P(\mathbf{x},t_1,t_2) \wedge P(\mathbf{x},t_3,t_4) \wedge \\ &(t_3 \leq t_2) \wedge (t_1 \leq t_4),\end{aligned} \tag{4}$$

which coalesces overlapping intervals (for example, $\mathit{Susp}(\mathit{adam})@[2,16]$ and $\mathit{Susp}(\mathit{adam})@[12,26]$ allow us to derive $\mathit{Susp}(\mathit{adam})@[2,26]$).

For instance when applying our translation to Rule (1), we obtain the following $\text{Datalog}_{\mathbb{Q}}$ rule:

$$\begin{aligned}&\mathit{Susp}(x,t^-,t^+ + 14) \leftarrow \\ &\quad \mathit{Trans}(x,y,t_1^-,t_1^+) \wedge \mathit{Trans}(y,x,t_2^-,t_2^+) \wedge \\ &\quad (t^- = \max\{t_1^-,t_2^-\}) \wedge (t^+ = \min\{t_1^+,t_2^+ + 5\}) \wedge \\ &\quad (t^- \leq t^+).\end{aligned} \tag{5}$$

In the rest of this section we will prove correctness of the translation and analyse how the materialisation steps for Π and $\mathcal{D}$ correspond to the materialisation steps of $\mathbf{tr}(\Pi)$ and $\mathbf{tr}(\mathcal{D})$. As we will show, establishing this relation is not straightforward, and poses several technical challenges.

Our analysis will exploit a stronger notion of correspondence between DatalogMTL and $\text{Datalog}_{\mathbb{Q}}$ interpretations than pointwise subsumptions from Definition 1. This is because application of rules does not only require that interpretations match on time points, but also on intervals. Hence, the following intervalwise subsumption will be useful.

Definition 3. A DatalogMTL interpretation $\mathfrak{I}$ is *intervalwise subsumed* by a $\text{Datalog}_{\mathbb{Q}}$ interpretation I, in symbols $\mathfrak{I} \leq_{\text{int}} I$, if for every metric fact $P(\mathbf{c})@\varrho$ satisfied by $\mathfrak{I}$ ($\mathfrak{I} \models P(\mathbf{c})@\varrho$) there exists an interval $[a,b] \supseteq \varrho$ such that $P(\mathbf{c},a,b) \in I$. On the other hand, I is *intervalwise subsumed* by $\mathfrak{I}$, in symbols $I \leq_{\text{int}} \mathfrak{I}$, if for every atom $P(\mathbf{c},a,b) \in I$, it holds that $\mathfrak{I} \models P(\mathbf{c})@[a,b]$. If $\mathfrak{I} \leq_{\text{int}} I$ and $I \leq_{\text{int}} \mathfrak{I}$, we write $\mathfrak{I} =_{\text{int}} I$.

It follows from Definitions 1 and 3 that intervalwise subsumption is a stronger notion than pointwise subsumption. Indeed, assume $\mathfrak{I} \leq_{\text{int}} I$ and consider an atom $P(\mathbf{c})@t$ such that $\mathfrak{I}, t \models P(\mathbf{c})$. Since $\mathfrak{I} \leq_{\text{int}} I$, there exists $[a, b]$ with $[t, t] \subseteq [a, b]$ such that $P(\mathbf{c}, a, b) \in I$. As a result, $\mathfrak{I} \leq_{\text{p}} I$. Likewise, $I \leq_{\text{int}} \mathfrak{I}$ implies $I \leq_{\text{p}} \mathfrak{I}$.

Recall that $\mathfrak{I}_\mathcal{D}$ is the minimal DatalogMTL interpretation satisfying the dataset $\mathcal{D}$. We observe that for each DatalogMTL dataset $\mathcal{D}$ we have $\mathbf{tr}(\mathcal{D}) \leq_{\text{int}} \mathfrak{I}_\mathcal{D}$. The opposite, however, is not always true.

Proposition 4. *For every DatalogMTL dataset $\mathcal{D}$, we have $\mathbf{tr}(\mathcal{D}) \leq_{int} \mathfrak{I}_\mathcal{D}$. However, the opposite subsumption does not always holds.*

Proof. By definition, $P(\mathbf{c}, a, b) \in \mathbf{tr}(\mathcal{D})$ implies that $P(\mathbf{c})@[a, b] \in \mathcal{D}$. Thus for every $t \in [a, b]$, $P(\mathbf{c})$ holds in $\mathfrak{I}_\mathcal{D}$ at time t, and so $\mathfrak{I} \models P(\mathbf{c})@[a, b]$.

The opposite subsumption, $\mathfrak{I}_\mathcal{D} \leq_{\text{int}} \mathbf{tr}(\mathcal{D})$, does not always hold. Indeed, consider a dataset $\mathcal{D} = \{Susp(adam)@[0, 20], Susp(adam)@[20, 40]\}$. We have $\mathfrak{I}_\mathcal{D} \models Susp(adam)@[0, 40]$, but $\mathbf{tr}(\mathcal{D}) = \{Susp(adam, 0, 20), Susp(adam, 20, 40)\}$ does not contain any atom $Susp(adam, a, b)$ with $[0, 40] \subseteq [a, b]$. □

As we show next, the interval subsumption does hold if we apply Π_{time} to $\mathbf{tr}(\mathcal{D})$ a sufficient number of times. In particular, we can show that $\mathfrak{I}_\mathcal{D} \leq_{\text{int}} T_{\Pi_{\text{time}}}^{\lceil \log(|\mathcal{D}|) \rceil}(\mathbf{tr}(\mathcal{D}))$, where we assume in the paper that log has base 2. The reason why we need to apply Π_{time} multiple times is that the dataset $\mathcal{D}$ may contain multiple adjacent or overlapping intervals for the same fact. Rule (4) merges two overlapping intervals, thus repeated applications of this rule perform a binary-style aggregation. Since each application halves the number of remaining intervals to be merged, all intervals contributing to a maximal union can be coalesced in logarithmically many steps. Formally, we obtain the following result.

Proposition 5. *For every DatalogMTL dataset $\mathcal{D}$, it holds that $\mathfrak{I}_\mathcal{D} =_{int} T_{\Pi_{time}}^{\lceil \log(|\mathcal{D}|) \rceil}(\mathbf{tr}(\mathcal{D}))$.*

Proof. Assume that $[a, b]$ is a subset-maximal interval such that $\mathfrak{I}_\mathcal{D} \models P(\mathbf{c})@[a, b]$, for some $P(\mathbf{c})$. Then, $[a, b]$ is a union of some intervals $[a_i, b_i]$, with $1 \leq i \leq n$, mentioned in $\mathcal{D}$. Hence, each $P(\mathbf{c}, a_i, b_i)$ belongs to $\mathbf{tr}(\mathcal{D})$. By applying Rule (4) at most $\lceil \log(n) \rceil$ times to $\mathbf{tr}(\mathcal{D})$, we can merge the intervals in a binary fashion and derive $P(\mathbf{c}, a, b)$. Since $n \leq |\mathcal{D}|$, we obtain $\mathfrak{I}_\mathcal{D} \leq_{\text{int}} T_{\Pi_{time}}^{\lceil \log(|\mathcal{D}|) \rceil}(\mathbf{tr}(\mathcal{D}))$. Conversely, for each $P(\mathbf{c}, a, b) \in T_{\Pi_{time}}^{\lceil \log(|\mathcal{D}|) \rceil}(\mathbf{tr}(\mathcal{D}))$, atom $P(\mathbf{c})$ holds in $\mathfrak{I}_\mathcal{D}$ at t in $[a, b]$, since every t belongs to some $[a_i, b_i]$. Thus $T_{\Pi_{time}}^{\lceil \log(|\mathcal{D}|) \rceil}(\mathbf{tr}(\mathcal{D})) \leq_{\text{int}} \mathfrak{I}_\mathcal{D}$. □

Propositions 4 and 5 provide us with a correspondence between $\mathcal{D}$ and $\mathbf{tr}(\mathcal{D})$. Next, we will analyse the correspondence between partial materialisations in DatalogMTL and Datalog$_\mathbb{Q}$. In particular, we will derive bounds on the number of the immediate consequence operator applications which guarantee intervalwise subsumptions between DatalogMTL and Datalog$_\mathbb{Q}$ interpretation. In the bounds below, we will use n_Π to represent the maximal number of metric atoms found in the rule bodies of rules in Π.

Theorem 6. *For a DatalogMTL dataset $\mathcal{D}$ and program Π*

$$T_\Pi(\mathfrak{I}_\mathcal{D}) \leq_{int} T_{\mathbf{tr}(\Pi)}^{\lceil \log(8|\mathcal{D}|^2(n_\Pi \cdot |\Pi|+1)) \rceil}(\mathbf{tr}(\mathcal{D})).$$

Proof sketch. Fix $P(\mathbf{c})@\varrho$ such that $T_\Pi(\mathfrak{I}_D) \models P(\mathbf{c})@\varrho$. We will show that $P(\mathbf{c}, a, b) \in T_{\mathbf{tr}(\Pi)}^K(\mathbf{tr}(D))$ for some $[a, b] \supseteq \varrho$, where $K = \lceil \log(8|\mathcal{D}|^2(n_\Pi|\Pi| + 1)) \rceil$.

By the definition of T_Π, interval ϱ is a union of finitely many intervals $\varrho_1, \ldots, \varrho_m$, such that each ϱ_i occurs explicitly in $\mathcal{D}$ or is an interval over which $P(\mathbf{c})$ is derived by a single rule $r \in \Pi$ applied to $\mathfrak{I}_\mathcal{D}$.

To compute a bound on m we determine how many endpoints can appear in these intervals. There are $2|\mathcal{D}|$ endpoints mentioned in $\mathcal{D}$. In the case of ϱ_i derived by T_Π, there are $|\Pi|$ rules, each with at most n_Π body atoms which gives rise to at most $4|\mathcal{D}| \cdot n_\Pi|\Pi|$ endpoints. In total, we obtain at most $4|\mathcal{D}|(n_\Pi|\Pi| + 1)$ endpoints, so $m \leq 4|\mathcal{D}|(n_\Pi|\Pi| + 1) - 1$.

We next show that for each ϱ_i there is $[a_i, b_i] \supseteq \varrho_i$ such that $P(\mathbf{c}, a_i, b_i) \in T_{\mathbf{tr}(\Pi)}^{K_1+1}(\mathbf{tr}(\mathcal{D}))$, where $K_1 = \lceil \log |\mathcal{D}| \rceil$. If ϱ_i is an interval appearing in $\mathcal{D}$, then the claim holds directly by Proposition 5, since $\Pi_{\text{time}} \subseteq \mathbf{tr}(\Pi)$. If ϱ_i is obtained by an application of a rule r with body $M_1 \wedge \cdots \wedge M_n$ satisfied at some ϱ_i' under a grounding ν, then $\mathfrak{I}_\mathcal{D} \models \nu(M_j)@\varrho_i'$ for each $j \leq n$. By Proposition 5, after K_1 applications of $T_{\mathbf{tr}(\Pi)}$ to $\mathbf{tr}(\mathcal{D})$ each translated body atom A_j from Definition 2 is derived. One further application of $T_{\mathbf{tr}(r)}$ then derives $P(\mathbf{c}, a_i, b_i)$ with $\varrho_i \subseteq [a_i, b_i]$, as required.

Since the atoms $P(\mathbf{c}, a_i, b_i)$ jointly cover ϱ, applying Rule (4) in a balanced binary fashion $\lceil \log m \rceil$ times merges them into a single $P(\mathbf{c}, a, b)$ with $\varrho \subseteq [a, b]$. This is obtained after at most $K_1 + 1 + \lceil \log m \rceil \leq K$ rule applications. □

For the opposite direction it is clear that the following subsumption holds.

Proposition 7. *For any DatalogMTL dataset $\mathcal{D}$ and any program Π it holds that*

$$T_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D})) \leq_{int} T_\Pi(\mathfrak{I}_\mathcal{D}).$$

As a consequence of the results established above, and the fact that the intervalwise subsumption implies pointwise subsumption, we obtain the following results.

Corollary 8. *For every DatalogMTL dataset $\mathcal{D}$ and any program Π it holds that*

$$T_{\mathbf{tr}(\Pi)}^\omega(\mathbf{tr}(\mathcal{D})) =_p T_\Pi^\omega(\mathfrak{I}_\mathcal{D}).$$

We can observe that the above holds for pointwise subsumptions, but not for the intervalwise subsumptions. Indeed, consider a DatalogMTL dataset $\mathcal{D}$ with a single fact $P(\mathbf{c})@0$ and program Π with a single rule $\boxplus_{[0,1]} P(\mathbf{x}) \leftarrow P(\mathbf{x})$. We obtain that $T_\Pi^\omega(\mathfrak{I}_\mathcal{D}) \models P(\mathbf{c})@[0, +\infty]$. However, $T_{\mathbf{tr}(\Pi)}^\omega(\mathbf{tr}(\mathcal{D}))$ cannot contain an atom $P(\mathbf{c}, 0, +\infty)$, since $+\infty$ cannot occur in any atom of $T_{\mathbf{tr}(\Pi)}^\omega(\mathbf{tr}(\mathcal{D}))$. As a result, $T_\Pi^\omega(\mathfrak{I}_\mathcal{D}) \not\leq_{\text{int}} T_{\mathbf{tr}(\Pi)}^\omega(\mathbf{tr}(\mathcal{D}))$.

The next example shows that for more than ω materialisation steps in DatalogMTL, bounding the corresponding amount of applications in Datalog$_{\mathbb{Q}}$ becomes infeasible. Consider a DatalogMTL dataset with a single fact $P(\mathbf{c})@[0,1]$, and a program Π with two rules: $\boxplus_{[0,1]}P(\mathbf{x}) \leftarrow P(\mathbf{x})$ and $Q(\mathbf{x}) \leftarrow \boxplus_{[0,+\infty]}P(\mathbf{x})$. We obtain that $T^{\omega}_{\Pi}(\mathfrak{I}_{\mathcal{D}}) \models P(\mathbf{c})@[0,+\infty]$, and $T^{\omega+1}_{\Pi}(\mathfrak{I}_{\mathcal{D}}) \models Q(\mathbf{c})@0$. As we have already argued, $P(\mathbf{c},0,\infty) \notin T^{\omega}_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D}))$. Therefore, $Q(\mathbf{c},0,0) \notin T^{\omega+1}_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D}))$.

Now we consider the notion of finite materialisability, that is, the possibility to materialise in a finite number of steps (within a finite number of immediate consequence operator applications). Our results on the bounds on the finite number of materialisation steps imply that the translation preserves finite materialisability: if $(\Pi,\mathcal{D})$ is finitely materialisable in DatalogMTL, then so is $(\mathbf{tr}(\Pi),\mathbf{tr}(\mathcal{D}))$ in Datalog$_{\mathbb{Q}}$, and vice versa.

Proposition 9. *For every DatalogMTL program* Π *and dataset* $\mathcal{D}$*, it holds that* $(\Pi,\mathcal{D})$ *is finitely materialisable in DatalogMTL if and only if* $(\mathbf{tr}(\Pi),\mathbf{tr}(\mathcal{D}))$ *is finitely materialisable in* Datalog$_{\mathbb{Q}}$.

We now consider the notion of consistency, for which we need to consider programs with $\bot$ in rule heads (programs without $\bot$ are trivially consistent). Our results on entailment allow us to obtain the following result regarding consistency.

Proposition 10. *For every DatalogMTL program* Π *and dataset* $\mathcal{D}$ *the following hold:*

- *if* $(\Pi,\mathcal{D})$ *is consistent in DatalogMTL, then* $\big(\mathbf{tr}(\Pi),\mathbf{tr}(\mathcal{D})\big)$ *is consistent in* Datalog$_{\mathbb{Q}}$*,*
- *if* $\big(\mathbf{tr}(\Pi),\mathbf{tr}(\mathcal{D})\big)$ *is consistent and* $\mathbf{tr}(\Pi)$ *is finitely materialisable in* Datalog$_{\mathbb{Q}}$*, then* $(\Pi,\mathcal{D})$ *is consistent in DatalogMTL.*

Proof sketch. We focus on the second item (the first one has a similar proof). Assume that $(\mathbf{tr}(\Pi),\mathbf{tr}(\mathcal{D}))$ is consistent and finitely materialisable. Hence, there exists $k \in \mathbb{N}$ such that $T^{k}_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D}))$ is the canonical interpretation. We define a DatalogMTL interpretation $\mathfrak{I}$ by setting, for each time point t and ground atom $P(\bar{c})$, $\mathfrak{I},t \models P(\bar{c})$ if and only if there exists an interval $[a,b]$ with $a \le t \le b$ such that $P(\bar{c},a,b) \in T^{k}_{\mathbf{tr}(\Pi)}(\mathbf{tr}(\mathcal{D}))$. We can show that $\mathfrak{I}$ is a model of Π and $\mathcal{D}$, so $(\Pi,\mathcal{D})$ is consistent as required. □

Notice that the second item of Proposition 10 requires the assumption of finite materialisability. Without this assumption, it may be the case that $\big(\mathbf{tr}(\Pi),\mathbf{tr}(\mathcal{D})\big)$ is consistent, but $(\Pi,\mathcal{D})$ is not. This happens, for example, if $\mathcal{D}$ has a single fact $P(\mathbf{c})@[0,1]$, whereas Π has two rules: $\boxplus_{[0,1]}P(\mathbf{x}) \leftarrow P(\mathbf{x})$ and $\bot \leftarrow \boxplus_{[0,+\infty]}P(\mathbf{x})$.

5 Implementation and Experiments

The aim of this section is to test practical viability of the translation-based approach, by comparing translated DatalogMTL programs executed with existing non-temporal engines with a state-of-the-art dedicated DatalogMTL reasoner. Our experiments focus on acyclic and finitely materialisable programs, which are commonly considered in current DatalogMTL implementations and applications.

We observe that our translation introduces predicates with arities increased by two, to represent interval endpoints. Such a translation can be run directly with solvers such as Nemo (Ivliev et al. 2024), which allow for predicates with arbitrary arities. However, some high-quality Datalog reasoners are based on the triple-based data format, which restricts the arity of allowed predicates to two. In order to be able to perform DatalogMTL reasoning in such reasoners, we will first adapt our translation to this setting.

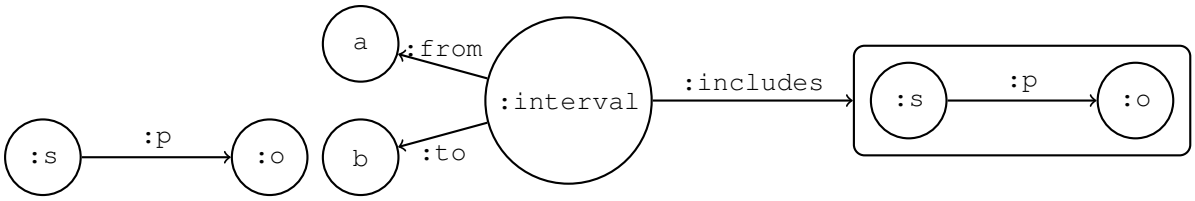
Translation to N3 The N3 format builds upon the RDF triple patterns, of the form `:s :p :o`, where s and o are the subject and object of predicate P. Atoms $P(s,o)$ thus have predicates P restricted to arity two. N3 introduces graph terms as a means to add metadata to a group of triples. We employ this structure to encode when an atom $P(s,o)$ holds. In order to adapt the DatalogMTL translation to this setting, we encode a temporal fact of the form $P(s,o)@[a,b]$ into N3 as

```
:interval :includes { :s :p :o }.
:interval :from :a.
:interval :to :b .
```

which is depicted in Figure 2b. Translation of DatalogMTL rules is likewise adapted to the new temporalized triple structure. Each rule body atom $P(s,o,t_i^-,t_i^+)$ is represented by three N3 triples, as described above. Both subject and object s,o and the temporal bounds ϱ_i^-,ϱ_i^+ are thereby encoded as variables, i.e. `?s`, `?o`, `?start_i` and `?end_i`. Numerical constraints from operators like $\boxplus$, $\boxminus$, $\mathcal{S}$, and $\mathcal{U}$ are expressed using `math:difference` and `math:sum` clauses. To construct the head interval, we use `math:min` and `math:max` to compute the maximum lower bound and minimum upper bound across all body atoms, adjusted by the temporal operator. A fresh interval node `?newInterval` is introduced, along with edges `:from ?start`, `:to ?end` and `:includes { :s :q :o }` for a head atom with predicate $Q(s,o)$.

Systems We test the translation-based approach for reasoning in DatalogMTL using three non-temporal reasoning engines, namely Datalog engines Nemo and Eyelet, and an N3 reasoner EYE. To evaluate practical applicability of this approach for reasoning in DatalogMTL, we ran translated Datalog$_{\mathbb{Q}}$ programs on Nemo and Eyelet, and translated N3 programs on EYE. As the baseline, we use the dedicated DatalogMTL reasoner MeTeoR.

Experiments In the experiments we have computed all facts entailed by a given pair of a DatalogMTL program and dataset (i.e. we compute the canonical interpretations).



(a) Standard RDF triple (b) Temporal N3 term

Figure 2: Graphic representation of temporalization of RDF triples

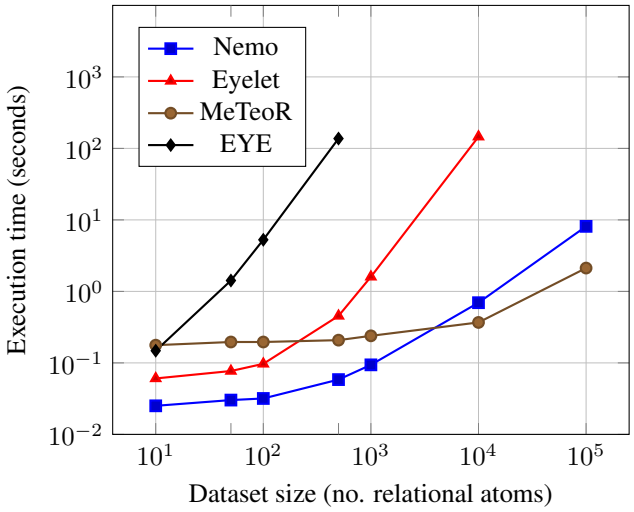


Figure 3: Average execution time on iTemporal benchmark

We ran experiments on synthetic benchmarks generated using iTemporal (Bellomarini et al. 2025), the only available generator producing both DatalogMTL programs and datasets. We generate datasets with varying number of relational atoms, (10, 50, 10^2, 500, 10^3, 10^4, 10^5) as proxy for dataset size used by iTemporal. Each dataset uses 10 predicate symbols. All datasets are acyclic and finitely materialisable.

Experiments were run on a Dell Latitude 5530 (Intel i7-1265U, 16GB RAM). Each configuration is repeated 5 times, preceded by a warm-up run to avoid initialization bias. Since applying our translations takes under 800ms even for the largest dataset, we excluded them from runtime comparisons. Runtimes depicted in Figure 3 include time for data import and materialisation, as this provides the most consistent basis for comparison across engines with differing execution models. All engines were verified to produce the same outputs, ensuring the correctness of the results. The full set of programs, datasets, and auxiliary files of our experiments are provided via a Github repository[1].

[1]https://github.com/MathijsvN/DatalogMTL_Embedding_Experiments

Results Our results, presented in Figure 3, show that Nemo outperforms MeTeoR, Eyelet, and EYE on small datasets. The performance gap between Nemo and MeTeoR narrows as dataset size increases, with Nemo surpassing MeTeoR for sizes above 10^4; at size 10^5, MeTeoR is on average 5.994 seconds faster (73.9% of Nemo's runtime), but this difference is smaller than Nemo's runtime variability (SD = 0.93). Overall, runtime variability is low for smaller datasets (SD < 0.04 s for all engines). For larger datasets, MeTeoR's variance remains stable (SD $\leq$ 0.13 s), while Nemo (SD up to 0.93 s), Eyelet (SD up to 18.93 s), and EYE (SD up to 2.57 s) show increasing variability with dataset size.

These results demonstrate that, under the considered assumptions, translation to standard Datalog provides a viable alternative to dedicated DatalogMTL reasoners, particularly for moderately sized datasets. Notably, the best performance is observed when using Datalog reasoners. Performance of reasoning on the N3 syntax increases rapidly with dataset size, as experiments only up to dataset size 500 could be obtained. The N3 format thus appears to introduce a substantial performance overhead.

6 Conclusion and Future Work

We have presented a translation from DatalogMTL into standard Datalog with arithmetic, which allows us to execute temporal reasoning tasks using mature, non-temporal reasoning engines. The results show that temporal reasoning in DatalogMTL can be effectively performed by translating programs into $\text{Datalog}_{\mathbb{Q}}$ and running standard Datalog reasoning. The theoretical analysis ensures that the translation preserves central semantic properties, including entailment, consistency, and finite materialisability.

Beyond the theoretical correspondence, our empirical evaluation highlights the practical viability of this translation-based method. For small datasets, translated DatalogMTL programs executed on Nemo outperform MeTeoR, a state-of-the-art specialised temporal reasoner. Even when performance is comparable rather than superior, the ability to rely on existing, professionally engineered Datalog platforms represents a major advantage in terms of maintainability, deployment readiness, and ecosystem integration.

As the next step we plan to scale up the empirical evaluation and to determine realistic use-cases where this approach can be applied. Additionally, further steps can be taken with regards to the theoretical aspect of the work. In particular it is interesting to explore the interaction of our translation with stratified negation (Cucala et al. 2021) and

incremental reasoning techniques (Zhao et al. 2026) in DatalogMTL. Furthermore, other stream reasoning languages such as LARS and RTEC can be considered for a similar mapping to non-temporal engines.

Acknowledgements

This research partially is funded by the FWO Travel Grant (Nr. V424925N) and FWO Project FRACTION (Nr. G086822N).

AI Use Disclosure

AI-based tools were used solely for editorial assistance, including improving readability and presentation. All technical content and conclusions are entirely the authors' own.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Anicic, D.; Rudolph, S.; Fodor, P.; and Stojanovic, N. 2012. Stream reasoning and complex event processing in etalis. *Semantic web* 3(4):397–407.

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

Artale, A.; Kontchakov, R.; Kovtunova, A.; Ryzhikov, V.; Wolter, F.; and Zakharyaschev, M. 2015. First-order rewritability of temporal ontology-mediated queries. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2706–2712.

Artikis, A.; Sergot, M.; and Paliouras, G. 2012. Run-time composite event recognition. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, 69–80.

Baader, F.; Borgwardt, S.; Koopmann, P.; Ozaki, A.; and Thost, V. 2017. Metric temporal description logics with interval-rigid names. In *Frontiers of Combining Systems*, 60–76.

Barbieri, D. F.; Braga, D.; Ceri, S.; Valle, E. D.; and Grossniklaus, M. 2010. C-SPARQL: a continuous query language for RDF data streams. *International Journal of Semantic Computing* 4(01):3–25.

Bazoobandi, H. R.; Beck, H.; and Urbani, J. 2017. Expressive stream reasoning with LASER. In *International Semantic Web Conference*, 87–103. Springer.

Beck, H.; Dao-Tran, M.; and Eiter, T. 2018. LARS: A logic-based framework for analytic reasoning over streams. *Artificial Intelligence* 261:16–70.

Beck, H. 2018. *Expressive rule-based stream reasoning*. Ph.D. Dissertation, Technische Universität Wien.

Bellomarini, L.; Blasi, L.; Nissl, M.; and Sallinger, E. 2022. The temporal Vadalog system. In *International Joint Conference on Rules and Reasoning*, 130–145.

Bellomarini, L.; Blasi, L.; Nissl, M.; and Sallinger, E. 2025. The Temporal Vadalog System: Temporal Datalog-Based Reasoning. *Theory and Practice of Logic Programming* 25(2):168–196.

Bellomarini, L.; Nissl, M.; and Sallinger, E. 2021a. Monotonic aggregation for temporal Datalog. *RuleML+ RR (Supplement)* 2956.

Bellomarini, L.; Nissl, M.; and Sallinger, E. 2021b. Query evaluation in DatalogMTL – taming infinite query results. https:/arxiv.org/abs/2109.10691. Accessed 2025-07-23.

Bellomarini, L.; Nissl, M.; and Sallinger, E. 2022. iTemporal: an extensible generator of temporal benchmarks. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2021–2033. IEEE.

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

Bienvenu, M.; Bourgaux, C.; and Khodadaditaghanaki, A. 2025. Inconsistency handling in DatalogMTL. https://arxiv.org/abs/2505.10394. Accessed 2025-07-22.

Brandt, S.; Kalayci, E. G.; Kontchakov, R.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2017. Ontology-based data access with a Horn fragment of metric temporal logic. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 1070–1076. AAAI Press.

Brandt, S.; Kalaycı, E. G.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2018. Querying Log Data with Metric Temporal Logic. *Journal of Artificial Intelligence Research* 62:829–877.

Chomicki, J., and Imielinski, T. 1988. Temporal deductive databases and infinite objects. In *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 61–73.

Colombo, A.; Bellomarini, L.; Ceri, S.; Laurenza, E.; et al. 2023. Smart derivative contracts in DatalogMTL. In *Proceedings of the 26th International Conference on Extending Database Technology, EDBT 2023*, 773–781.

Colombo, A.; Baldazzi, T.; Bellomarini, L.; Gentili, A.; Sallinger, E.; et al. 2024. LLM-based DatalogMTL modelling of MiCAR-compliant crypto-assets markets. In *Proceedings of the 5th International Workshop on the Resurgence of Datalog in Academia and Industry, Datalog-2.0 2024*, 17–22. CEUR-WS.

Cucala, D. J. T.; Wałęga, P. A.; Cuenca Grau, B.; and Kostylev, E. V. 2021. Stratified negation in Datalog with metric temporal operators. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 6488–6495.

De Brouwer, M.; Steenwinckel, B.; Fang, Z.; Stojchevska, M.; Bonte, P.; De Turck, F.; Van Hoecke, S.; and Ongenae, F. 2023. Context-aware & privacy-preserving homecare monitoring through adaptive query derivation for iot data streams with divide. *Semantic Web Journal*.

De Meester, B.; Arndt, D.; Bonte, P.; Bhatti, J.; Dereuddre, W.; Verborgh, R.; Ongenae, F.; De Turck, F.; Mannens, E.; and Van de Walle, R. 2015. Event-driven rule-based reasoning using eye. In *Joint Proceedings of the 1st Joint International Workshop on Semantic Sensor Networks and Terra Cognita (SSN-TC 2015) and the 4th International Workshop on Ordering and Reasoning (OrdRing 2015) co-*

located with the 14th International Semantic Web Conference (ISWC 2015), volume 1488, 75–86. CEUR Workshop Proceedings.

Dell'Aglio, D.; Della Valle, E.; Calbimonte, J.-P.; and Corcho, O. 2014. Rsp-ql semantics: A unifying query model to explain heterogeneity of rdf stream processing systems. *International Journal on Semantic Web and Information Systems (IJSWIS)* 10(4):17–44.

Gutiérrez-Basulto, V.; Jung, J. C.; and Ozaki, A. 2016. On metric temporal description logics. In *22nd European Conference on Artificial Intelligence*, 837–845.

Ivliev, A.; Gerlach, L.; Meusel, S.; Steinberg, J.; and Krötzsch, M. 2024. Nemo: your friendly and versatile rule reasoning toolkit. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning*, 743–754.

Kalaycı, E. G.; Xiao, G.; Ryzhikov, V.; Kalayci, T. E.; and Calvanese, D. 2018. Ontop-temporal: a tool for ontology-based query answering over temporal data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1927–1930.

Kontchakov, R.; Pandolfo, L.; Pulina, L.; Ryzhikov, V.; and Zakharyaschev, M. 2016. Temporal and spatial OBDA with many-dimensional Halpern-Shoham logic. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1160–1166.

Lanzinger, M.; Nissl, M.; Sallinger, E.; and Wałęga, P. A. 2023. Temporal Datalog with existential quantification. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 3277–3285.

Lécué, F. 2012. Diagnosing changes in an ontology stream: A DL reasoning approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 80–86.

Mori, M.; Papotti, P.; Bellomarini, L.; and Giudice, O. 2022. Neural machine translation for fact-checking temporal claims. In *Proceedings of the Fifth Fact Extraction and VERification Workshop*, 78–82.

Nenov, Y.; Piro, R.; Motik, B.; Horrocks, I.; Wu, Z.; and Banerjee, J. 2015. RDFox: A highly-scalable RDF store. In *International Semantic Web Conference*, 3–20. Springer.

Nickles, M., and Mileo, A. 2014. Web stream reasoning using probabilistic answer set programming. In *International Conference on Web Reasoning and Rule Systems*, 197–205. Springer.

Nissl, M., and Sallinger, E. 2022a. Modelling smart contracts with DatalogMTL. In *Proceedings of the Workshops of the EDBT/ICDT*.

Nissl, M., and Sallinger, E. 2022b. Towards bridging traditional and smart contracts with Datalog-based languages. In *Datalog*, 68–82.

Pham, T.-L.; Ali, M. I.; and Mileo, A. 2019. C-ASP: continuous asp-based reasoning over RDF streams. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, 45–50. Springer.

Phuoc, D. L. 2013. *A native and adaptive approach for linked stream data processing*. Ph.D. Dissertation, NUI Galway.

Raheb, K. E.; Mailis, T.; Ryzhikov, V.; Papapetrou, N.; and Ioannidis, Y. E. 2017. Balonse: Temporal aspects of dance movement and its ontological representation. In *European Semantic Web Conference*, 49–64.

Ryzhikov, V.; Wałęga, P. A.; and Zakharyaschev, M. 2019. Data complexity and rewritability of ontology-mediated queries in metric temporal logic under the event-based semantics. In *International Joint Conferences on Artificial Intelligence*, 1851–1857.

Thost, V. 2018. Metric temporal extensions of DL-Lite and interval-rigid names. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 665–666.

Wałęga, P. A.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2019. DatalogMTL: computational complexity and expressive power. In *International Joint Conferences on Artificial Intelligence*, 1886–1892.

Wałęga, P. A.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2020. DatalogMTL over the integer timeline. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 768–777.

Wałęga, P. A.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2021. Tractable fragments of Datalog with metric temporal operators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1919–1925.

Wałęga, P. A.; Zawidzki, M.; Wang, D.; and Cuenca Grau, B. 2023. Materialisation-based reasoning in DatalogMTL with bounded intervals. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 6566–6574.

Wałęga, P. A.; Zawidzki, M.; and Cuenca Grau, B. 2021. Finitely materialisable Datalog programs with metric temporal operators. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 18, 619–628.

Wałęga, P.; Zawidzki, M.; and Cuenca Grau, B. 2023. Finite materialisability of Datalog programs with metric temporal operators. *Journal of Artificial Intelligence Research* 76:471–521.

Wałęga, P. A.; Tena Cucala, D. J.; Kostylev, E. V.; and Cuenca Grau, B. 2021. DatalogMTL with negation under stable models semantics. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 609–618.

Wang, D.; Hu, P.; Wałęga, P. A.; and Cuenca Grau, B. 2022. Meteor: Practical reasoning in Datalog with metric temporal operators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 5906–5913.

Wang, D.; Cuenca Grau, B.; Wałęga, P. A.; and Hu, P. 2025a. Practical reasoning in datalogmtl. *Theory and Practice of Logic Programming* 25(2):225–255.

Wang, S.; Zhao, K.; Wei, D.; Wałęga, P. A.; Wang, D.; Cai, H.; and Hu, P. 2025b. Goal-driven reasoning in DatalogMTL with magic sets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 15203–15211.

Wang, D.; Wałęga, P. A.; and Cuenca Grau, B. 2024. MTLearn: extracting temporal rules using Datalog rule learners. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 21, 962–973.

Zhao, K.; Chen, D.; Wang, S.; and Hu, P. 2026. Incremental maintenance of datalogmtl materialisations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, 19467–19476.