

Temporal Datalog in Academia and Industry: The Temporal Vadalog System

Luigi Bellomarini^{1,*}, Livia Blasi^{2,1,*}, Eleonora Laurenza^{2,*}, Markus Nissl^{2,*} and Emanuel Sallinger^{2,*}

¹Central Bank of Italy

²TU Wien

Abstract

In this work, we present Temporal Vadalog, a Temporal Datalog-based system. Following the motto of Datalog 2.0 “the resurgence of Datalog in Academia and Industry”, we focus on clear examples of the use of Temporal Vadalog in financial application scenarios. This includes time series operations such as shifting, resampling, moving averages and “stock and flow” operations, which we illustrate through concrete temporal implementations.

Keywords

Datalog, Temporal Datalog, DatalogMTL, Temporal Vadalog, Finance and Economics

1. Introduction

In this paper, we present Temporal Vadalog, a Temporal Datalog-based system. Following the motto of Datalog 2.0 “the resurgence of Datalog in Academia and Industry” – we focus not so much on the Temporal Vadalog system itself, but on effective uses of the Temporal Vadalog language and its application in settings such as in finance and economics together with the Central Bank of Italy. This paper is based on the 2025 TPLP paper [1].

The Resurgence of Datalog. In recent years, the Datalog language [2] has been experiencing renewed interest, thanks to the growing adoption for *Knowledge Representation and Reasoning* applications [3]. The additional requirements introduced by reasoning spawned prolific research towards new extensions of the language to support advanced features such as existential quantification and aggregation. These extensions are frequently shared under the common name of Datalog[±] [4] and include languages, technically, “fragments”, that exhibit a very good trade-off between computational complexity and expressive power, such as Warded and Shy Datalog[±] [5].

As the new logical languages flourish, the expectations for declarative, explainable expressivity of complex domains, jointly with feasible computation, are rekindling the idea of using deductive languages in practical applications, with modern systems [6] based on Datalog in production settings, often in combination with knowledge-based AI models, such as *knowledge graphs* [7].

Temporal Datalog. With so many real-world adoptions, the temporal perspective is increasingly becoming a first-class requirement of Datalog [8]. In fact, with the pervasive constant and voluminous streams of data, we are witnessing that temporal reasoning—the ability to reason about events and intervals in time—is becoming of the essence in many fields, from healthcare to finance, robotics to transportation. We can look at a plethora of use cases: the longitudinal studies of anticancer treatments to explain and prevent adverse events [9], the analysis and prediction of stock market time series [10], the tracking of objects for unmanned vehicles [11], the processing of IoT devices interactions [12], and the exploitation of log data [13].

Datalog 2.0 2026

*Corresponding authors.

✉ luigi.bellomarini@bancaditalia.it (L. Bellomarini); livia.blasi@tuwien.ac.at (L. Blasi); eleonora.laurenza@tuwien.ac.at (E. Laurenza); markus.nissl@tuwien.ac.at (M. Nissl); emanuel.sallinger@tuwien.ac.at (E. Sallinger)

0000-0001-6863-0162 (L. Bellomarini); 0000-0003-0701-1688 (L. Blasi); 0000-0002-2786-8163 (E. Laurenza); 0000-0001-8196-5688 (M. Nissl); 0000-0001-7441-129X (E. Sallinger)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

The proposal for an extension of Datalog with *Metric Temporal Logic* operators, or DatalogMTL [13], shows an extremely promising impulse towards a declarative language with sufficient expressive power to handle reasoning settings, such as recursion, while offering time awareness within a computationally effective framework.

To present the ideas at the foundation of Temporal Vadalog, let us start introducing DatalogMTL with a financial use case example from our work with the Bank of Italy.

Changes in strategically important companies may face government scrutiny to uphold regulatory standards, focusing on both the companies and their new shareholders. We describe the scenario by a database $\mathcal{D}$ of facts about company shares and the following set Π of DatalogMTL rules.

$$\begin{aligned} & \boxplus_{[0,1]} \text{significantShare}(X, Y), \\ & \neg\Diamond_{[0,1]} \text{significantShare}(X, Y) \rightarrow \text{significantOwner}(X, Y) \end{aligned} \quad (1)$$

$$\begin{aligned} & \text{watchCompany}(Y), \text{significantOwner}(X, Y), \\ & \text{connected}(X, Z) \rightarrow \text{watchCompany}(Z) \end{aligned} \quad (2)$$

We assume the reader is familiar with Datalog rule syntax: the left-hand side of the rule (the body) is the implication premise and is composed of a logical conjunction of predicates over terms, i.e., constants or variables; the right-hand side (the head) is the implication conclusion. In the rules, we find predicates from the relational schema of $\mathcal{D}$. In particular, *significantShare* holds whenever X retains a high number of shares of Y ; *significantOwner* is then a derived, namely, “*intensional*” predicate, obtained by Rule 1.

Then, by Rule 2, whenever Y is interested in a *watchCompany* fact to indicate that the authority deems that company of some strategic relevance, given that X is a significant owner of Y , then all companies Z *connected* to X will be watched over as well. Neglecting for a moment the temporal angle of Π , its semantics is straightforward: whenever the premise of one rule holds with respect to $\mathcal{D}$, then $\mathcal{D}$ is augmented with new facts to satisfy the implication if it does not already hold.

For instance, if $\mathcal{D}$ contains the facts *watchCompany*(*ACME*), *significantOwner*(*A*, *ACME*), *connected*(*A*, *EMCA*), then the new fact *watchCompany*(*EMCA*) will be added to $\mathcal{D}$. Here, the temporal perspective is introduced by the $\boxplus$ and $\Diamond$ operators. When prefixed with $\boxplus$, a predicate holds, only if the predicate itself continuously holds in the interval $[t, t + 1]$, if Rule 1 is evaluated at t in the future, as indicated by the operator internal in the index.

The $\Diamond$ operator modifies a predicate’s semantics so that it holds if the predicate itself holds at least once in the past interval $[t - 1, t]$, if Rule 1 is evaluated at t . Clearly, in a temporal interpretation of Datalog, all facts, including the “*extensional*” ones of $\mathcal{D}$, are time-annotated. In the example, Rule 1 fires only if X , who has never been a significant shareholder for Y in the past, becomes such in the future.

In this short paper, we deliberately do not focus on extended syntax and semantics definitions to be able to showcase some of the core language features regarding temporal operations – which we will focus on in the following section. We refer the interested reader to [1] for syntax and semantics details, as well as further details on the Temporal Vadalog system.

2. Time Series in Temporal Vadalog

Time series data is a sequence of data points, collected over time, that measure the performance of a particular variable. For example, we may have time series for stock prices or inflation expectations in finance and economics, humidity readings in domotics, and all the relevant vital measurements for healthcare.

Time series are typically made of sequences of evenly-spaced punctual temporal intervals, in which case they are considered *regular*; otherwise, they are called *irregular* time series. The distance between intervals defines the frequency (or time resolution).

To understand how to work with time series, we need a clear overview of the operators that can be applied and how they manipulate and transform data to detect and measure patterns and trends. Let us go through an introduction to the common time series operators, and we will later explore their implementation in Temporal Vadalog.

Shifting. The shifting operator moves the time index of the data, allowing to compare data points from different periods (e.g., the same month in different years).

Rolling. The rolling operator calculates a moving time window (rolling window), typically to apply statistical metrics to smooth out short-term data fluctuations.

Resampling. The resampling operator transforms the time series by changing its frequency, by lowering it (*downsampling*) or by increasing it (*upsampling*).

Moving Averages. Moving averages use rolling windows to smooth data and filter noise and fluctuations. Types include: a) *Simple Moving Average*, where all points have equal weight, and the *Centered Moving Average* variant; b) *Exponential Moving Average*, where recent points have more weight.

Stock and flow. Measurements in time series can be stocks, quantities existing at a specific time, or flows, variables measured over an interval of time, comparable to the rate or speed, and can be derived from stocks (and vice versa).

Seasonal decomposition. The seasonal decomposition operator splits a time series into its trend, seasonal, and residual components, which helps identify patterns and trends.

Time series analysis is vital across many sectors, using historical data to identify trends, detect anomalies, predict events, and inform decisions to improve efficiency, reduce costs, and allocate resources. While many Time Series Database (TSDB) systems exist, considering time series analysis relevance in data science, we want to explore how to handle it with a more general-purpose system like Temporal Vadalog, allowing us to use common operators for time series in the broader scope of reasoning applied, for instance, to KGs, specifically exploiting its characteristics of explainability and context-awareness, especially important for domains dealing with sensitive data and particular needs.

In this section, we will present how the Temporal Vadalog System can be used to reason over time series, by showing how many of the core functions of time series databases are easily and immediately available by using DatalogMTL operators, monotonic temporal aggregations and arithmetic operations. Since we focus on regular time series, the examples in the following sections are intended to be used in discrete time (e.g., timestamps).

2.1. Basic operations

Temporal Vadalog natively handles the *sum* and other arithmetical operations over the same time intervals. This section covers other basic operations that deal with time series: shifting, rolling, and resampling. We will proceed informally by example, as the temporal operators' semantics has already been introduced.

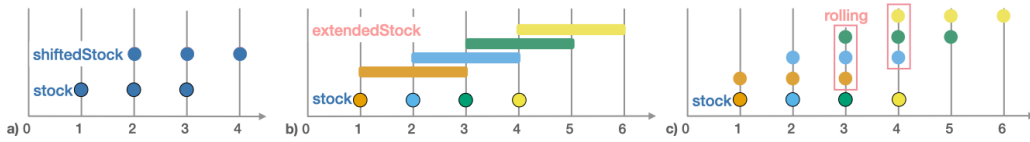


Figure 1: Basic time series operators in Temporal Vadalog: a) Shifting by 1; b) Rolling operator with $n = 3$; c) Join of the extended intervals with the original ones

Shifting. Shifting re-positions the time points of a time series by adding a delay, achieved through the box or diamond operator by stating a fact holds at t if it held at some past time point. The example adds a lag of one unit of time to the stock facts, where $stock(X, Value)$ represents any time series X with value $Value$. Figure 1a shows the shifting operation over a discrete timeline.

$$\Diamond_{[1,1]} stock(X, Value) \rightarrow shiftedStock(X, Value) \quad (11)$$

Rolling. Rolling defines a fixed-size window that slides through a time series one data point at a time, allowing for the computation of statistics over the data points within the window. A rolling window of size n is achieved by extending each data point's interval by $[0, n)$ with the diamond operator (Figure 1b). The time series is later joined with the original one, applying the full window's data points to each specific interval for further operations (Figure 1c where at $t = 3, 4$ we have the $n = 3$ rolling windows.)

$$\diamond_{[0,n)} stock(X, Value) \rightarrow extended(X, Value) \quad (12)$$

$$stock(X, Value), extended(X, Roll) \rightarrow rolling(X, Roll) \quad (13)$$

Resampling. The resampling of a time series changes its time resolution. Working with different time resolutions (e.g., one monthly series and one daily) may require this change in both directions: *downsampling*, lowering the frequency of the data (e.g. daily to monthly), and *upsampling*, increasing the frequency (e.g. monthly to daily).

Downsampling. In Temporal Vatalog we can achieve *downsampling* by using the aggregation operator $\triangle_{unit}$, where *unit* is the frequency we want to transform our time series into. This operator was introduced in a previous work [14].

$$\triangle_{month} dailyStock(X, Value) \rightarrow monthlyStocks(X, Value) \quad (14)$$

The set of facts that were valid daily will be now valid over the entire month. To obtain the final value for the monthly data point, we can use arbitrary aggregations, depending on the domain (e.g., arithmetic mean, minimum, maximum, etc.)

Upsampling. We can convert a time series to a higher frequency by using the temporal join with a series with a higher frequency that we can create on the spot with a diamond operator. In the following example, we will convert a *weeklyStock*(Company, Value, WeekStart), where *WeekStart* is expressed in a day unit, into a daily time series.

$$\diamond_{[1,1]} dailySeries(X), Y = X + 1 \rightarrow dailySeries(Y) \quad (15)$$

$$\diamond_{[0,7)} weeklyStock(C, V, W) \rightarrow nextWeek(C, V, W) \quad (16)$$

$$dailySeries(Z), weeklyStock(C, V_1, W_1), \\ nextWeek(C, V_2, W_2), W = W_1 + \frac{(V-V_1)(W_2-W_1)}{(V_2-V_1)} \rightarrow upsampling(C, W) \quad (17)$$

Note that Rule 15 generates an infinite CHASE of intervals, which requires us to employ a mechanism for terminating the generations like the strategies discussed in [1].

2.2. Moving Averages

Moving averages use the rolling window described above to calculate different types of averages, typically to smooth out fluctuations and noise.

Simple Moving Average (SMA). In this moving average, each data point contained in the window has the same weight in the calculation of the average. Hence, we compute first the rolling window and then derive the arithmetic mean of the data points (Figure 2a).

$$rolling(X, Y), Avg = avg(Y) \rightarrow sma(X, Avg) \quad (18)$$

Exponential Moving Average (EMA). With the EMA operator we give more importance to the more recent data points. The first value is calculated through the simple moving average. We omit this from the example for conciseness. The following values are calculated as a sum (Rule 21) of the current data point multiplied by the parameter k (where $k = 2/(1 + n)$, n being the size of the window) (Rule 20) and, using recursion, the previous moving average multiplied by $1 - k$ (Rule 19).

$$\diamond_{[1,1]} ema(X), Y = X * (1 - k) \rightarrow emaInput(Y) \quad (19)$$

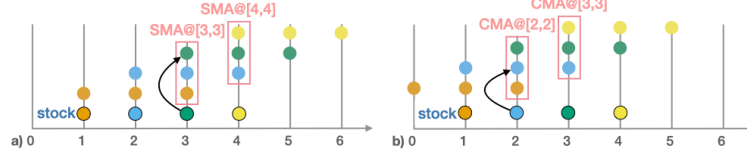


Figure 2: From Simple moving average (a) to Centered moving average (b), with $n = 3$.

$$stock(X, V), Y = X * k \rightarrow emaInput(Y) \quad (20)$$

$$emaInput(X), Y = sum(X) \rightarrow ema(Y) \quad (21)$$

Centered Moving Average (CMA). In the SMA operator, a window of n data points will yield a value at the right extremity of that window; in the Centered Moving Average (CMA), instead, the data point of reference for the moving average is the data point at the center of the window. For any odd n window, this is obtained by calculating the SMA and then shifting it by $n/2$, as shown by Figure 2a-b.

$$\Diamond_{[0, n/2]} sma(X, Y) \rightarrow cma(X, Y) \quad (22)$$

2.3. Stock and Flow

Performing mutual conversion between stock and flow is a helpful feature in multiple contexts. Transformation operators, in continuous time, can be computed through derivatives and integrals, while in discrete time one usually considers the difference between two time points. We focus on the discrete version, as the discrete (integer) timelines are the usual implementation choice in time series databases [15].

Stock to Flow (discrete). This operator calculates the flow as the difference between the stock at t and $t - 1$. To do so in Temporal Vadalog, we shift the time series by $[1, 1]$ and compute the difference between the values from the original and shifted time series.

$$\Diamond_{[1, 1]} stock(X, V) \rightarrow shiftedStock(X, V) \quad (23)$$

$$stock(X, V1), shiftedStock(X, V2), V = V1 - V2 \rightarrow stockToFlow(X, V) \quad (24)$$

Flow to Stock (discrete). This operator, on the other hand, calculates the stock as the cumulative sum of flows from the beginning of the time series up to the current data point. To do so in Temporal Vadalog, we extend the flow time series to the interval $[0, T]$, where T is the last interval, and then we compute the sum over each data point.

$$\Diamond_{[0, T]} flow(X, V) \rightarrow rolledFlow(X, V) \quad (25)$$

$$rolledFlow(X, V1), V = sum(V1) \rightarrow stock(X, V) \quad (26)$$

3. Conclusion

In this paper, we highlighted the importance of time series operators in Temporal Datalog, concretely the Temporal Vadalog language – and illustrated how to concretely implement typical operations needed in financial and economic analyses such as at the Central Bank of Italy. Further details can be found in [1, 16, 17, 18, 14], including benchmark datasets and generators [19].

As future work, there is a lot of potential in the area of Temporal Datalog, including broadening the application to further domains [20] such as enterprise architecture [21] and smart contracts [22], and continued work on reasoners in the area, especially in neuro-symbolic combinations such as with embeddings [23] and beyond first-order logic [24, 25].

Declaration on Generative AI

For the preparation of this work, the authors used no Generative AI tools.

Acknowledgements. The work on this paper was supported by the Vienna Science and Technology Fund (WWTF) [Grant ID: 10.47379/VRG18013, 10.47379/ICT25032, 10.47379/NXT22018, 10.47379/ICT2201, 10.47379/DCDH001] and the Austrian Science Fund (FWF) 10.55776/COE12."

References

- [1] L. Bellomarini, L. Blasi, M. Nissl, E. Sallinger, The temporal vatalog system: Temporal datalog-based reasoning, *Theory Pract. Log. Program.* 25 (2025) 168–196. URL: <https://doi.org/10.1017/s1471068425000018>. doi:10.1017/S1471068425000018.
- [2] S. Ceri, G. Gottlob, L. Tanca, What you always wanted to know about datalog (and never dared to ask), *IEEE Trans. Knowl. Data Eng.* 1 (1989) 146–166. URL: <https://doi.org/10.1109/69.43410>. doi:10.1109/69.43410.
- [3] G. Gottlob, Adventures with datalog: Walking the thin line between theory and practice, in: *AI*IA*, volume 13796 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 489–500.
- [4] A. Cali, G. Gottlob, T. Lukasiewicz, A general datalog-based framework for tractable query answering over ontologies, *J. Web Semant.* 14 (2012) 57–83. URL: <https://doi.org/10.1016/j.websem.2012.03.001>. doi:10.1016/J.WEBSEM.2012.03.001.
- [5] T. Baldazzi, L. Bellomarini, M. Favorito, E. Sallinger, On the relationship between shy and warded datalog+/-, in: *KR*, 2022. URL: <https://proceedings.kr.org/2022/39/>.
- [6] N. Leone, C. Allocca, M. Alviano, F. Calimeri, C. Civili, R. Costabile, A. Fiorentino, D. Fuscà, S. Germano, G. Labocchetta, B. Cuteri, M. Manna, S. Perri, K. Reale, F. Ricca, P. Veltri, J. Zangari, Enhancing DLV for large-scale reasoning, in: *LPNMR*, volume 11481, 2019, pp. 312–325.
- [7] L. Bellomarini, D. Fakhoury, G. Gottlob, E. Sallinger, Knowledge graphs and enterprise AI: the promise of an enabling technology, in: *ICDE*, IEEE, 2019, pp. 26–37. URL: <https://doi.org/10.1109/ICDE.2019.00011>. doi:10.1109/ICDE.2019.00011.
- [8] D. Wang, P. Hu, P. A. Walega, B. C. Grau, Meteor: Practical reasoning in datalog with metric temporal operators, in: *AAAI*, AAAI Press, 2022, pp. 5906–5913.
- [9] M. Ma, K. Lee, Y. Mai, C. Gilman, Z. Liu, M. Zhang, M. Li, A. Redfern, T. Mullaney, T. Prentice, P. McDonagh, Q. Pan, R. Chen, E. Schadt, X. Wang, Extracting longitudinal anticancer treatments at scale using deep natural language processing and temporal reasoning., *Journal of Clinical Oncology* 39 (2021) e18747–e18747.
- [10] P. A. Walega, M. Kaminski, B. Cuenca Grau, Reasoning over streaming data in metric temporal datalog, in: *AAAI*, 2019, pp. 3092–3099.
- [11] F. Lin, C. Fu, Y. He, W. Xiong, F. Li, Recf: Exploiting response reasoning for correlation filters in real-time uav tracking, *IEEE Trans. Intell. Transp. Syst.* 23 (2022). doi:10.1109/TITS.2021.3094654.
- [12] P. A. Walega, B. Cuenca Grau, M. Kaminski, E. V. Kostylev, Datalogmtl: Computational complexity and expressive power, in: *IJCAI*, 2019, pp. 1886–1892.
- [13] S. Brandt, E. G. Kalayci, V. Ryzhikov, G. Xiao, M. Zakharyashev, Querying log data with metric temporal logic, *J. Artif. Intell. Res.* 62 (2018) 829–877.
- [14] L. Bellomarini, M. Nissl, E. Sallinger, Monotonic aggregation for temporal datalog, in: *RuleML+RR*, 2021.
- [15] C. E. Dyreson, Chronon, in: *Encyclopedia of Database Systems* (2nd ed.), Springer, 2018.
- [16] L. Bellomarini, L. Blasi, M. Nissl, E. Sallinger, The temporal vatalog system (short paper), in: *TIME*, 2025.
- [17] M. Lanzinger, M. Nissl, E. Sallinger, P. A. Walega, Temporal datalog with existential quantification, in: *IJCAI*, 2023.
- [18] L. Bellomarini, L. Blasi, M. Nissl, E. Sallinger, The temporal vatalog system, in: *RuleML+RR*, 2022.

- [19] L. Bellomarini, M. Nissl, E. Sallinger, itemporal: An extensible generator of temporal benchmarks, in: ICDE, 2022.
- [20] L. Bellomarini, R. R. Fayzrakhmanov, G. Gottlob, A. Kravchenko, E. Laurenza, Y. Nenov, S. Reissfelder, E. Sallinger, E. Sherkhonov, S. Vahdati, L. Wu, Data science with vadalog: Knowledge graphs with machine learning and reasoning in practice, *Future Gener. Comput. Syst.* 129 (2022) 407–422.
- [21] P. Glaser, S. J. Ali, E. Sallinger, D. Bork, Model-based construction of enterprise architecture knowledge graphs (extended abstract), in: EMISA, 2023.
- [22] M. Nissl, E. Sallinger, Modelling smart contracts with datalogmtl, in: EcoFinKG, 2022.
- [23] L. Bellomarini, E. Sallinger, S. Vahdati, Reasoning in knowledge graphs: An embeddings spotlight, in: *Knowledge Graphs and Big Data Processing*, volume 12072 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 87–101.
- [24] I. Feinerer, R. Pichler, E. Sallinger, V. Savenkov, On the undecidability of the equivalence of second-order tuple generating dependencies, *Inf. Syst.* 48 (2015) 113–129.
- [25] P. G. Kolaitis, R. Pichler, E. Sallinger, V. Savenkov, Nested dependencies: structure and reasoning, in: PODS, ACM, 2014, pp. 176–187.