Towards Explainable Sequential Learning[*]

Giacomo Bergami[1], Emma Packer[2], Kirsty Scott[2], and Silvia Del Din[2,3]

[1] School of Computing, Newcastle University
giacomo.bergami@newcastle.ac.uk
[2] Translational and Clinical Research Institute, Faculty of Medical Sciences, Newcastle University
{e.packer,kirsty.scott-singer}@newcastle.ac.uk
[3] National Institute for Health and Care Research (NIHR), Newcastle Biomedical Research Centre
(BRC), Newcastle University and The Newcastle upon Tyne Hospitals NHS Foundation Trust
silvia.del-din@newcastle.ac.uk

Abstract. This paper offers a hybridly explainable temporal data processing pipeline, DataFul Explainable MultivariatE coRrelatIonal Temporal Artificial inTElligence (EMeriTAte+DF), bridging numerical-driven temporal data classification with an event-based one through verified artificial intelligence principles, enabling human-explainable results. This was possible through a preliminary *a posteriori* explainable phase describing the numerical input data in terms of concurrent constituents with numerical payloads. This further required extending the event-based literature to design specification mining algorithms supporting concurrent constituents. Our previous and current solutions outperform state-of-the-art algorithms for multivariate time series classifications over four dataset considered in the present paper, thus showcasing the effectiveness of the proposed methodology premiering the extraction of explainable correlations across Multivariate Time Series (MTS) dimensions with dataful features.

Keywords: Verified AI; eXplainable AI (XAI); polyadic logs; Data Trends; Poly-DECLARE; Multivariate Time Series Classification

1.   Introduction

The current literature on temporal data analysis is divided into two main branches: one, mainly considering numerical fluctuations in different dimensions as Multivariate Time Series (MTS) [18,32,23,43], and the other focusing on a characterization as a linear sequence of discrete events, be they durative [3,37,5] or pointwise [17,11]. The first class denotes the classification task as Multivariate Time Series Classification (MTSC), which requires capturing behavioural correlations between different dimensions [38]. Cross-dimension correlations can be represented via convolutions [43] or kernels that potentially squash such correlations to a single numerical value [18]. Despite the possibility of doing so, the explanations thus obtained are not easily understandable by humans. Furthermore, recent results have shown that these solutions cannot capture correlations when considering complex and multifactorial problems, such as clinical ones, and give results with low accuracy [16]. These considerations require us to solve the problem from a different angle.

---

[*] This journal paper constitutes a major extension of our previous work [16].

Recent approaches are starting to bridge the numerical with the event-driven characterization of temporal data by clearly showing the possibility of discretizing Data Trends (DT) occurring into durative constituents [24]. Notwithstanding this, these events cannot summarize numerical information precisely characterizing the trend of choice, thus only characterizing growth, variability, or decrease patterns without providing any further numerical references to better describe the event of choice. This demands a more general representation of DT patterns, which should also encompass numerical features, albeit summarized (e.g. 22 CAnonical Time-series CHaracteristics (catch22) [30]), to characterize the underlying data from a more data-driven perspective.

Within the second branch, Business Process Management [1,17] enables, through declarative process mining, to temporally correlate single pointwise events through a human-readable data description while bridging across different *activity labels* (distinguishing different action/event types) occurring. This literature premiered the extraction of declarative specifications from event-based data while enriching information concerning temporal correlations with additional numerical [27] and categorical [14] data predicates. Despite the inherent ability of such techniques to provide a human-explainable characterization of the data, the former techniques cannot be straightforwardly applied to solve the MTSC problem, even after DT-discretization. *First*, despite the possibility of representing temporal specifications as concurrent ones [1], these are always interpreted as a temporally-ordered succession of non-concurrent and non-overlapping events [15], thus invalidating the possibility of directly exploiting the same algorithms for capturing multiple events co-occurring at the same time across dimensions. In fact, despite the possibility for the current XES [2] standard to express durative events, this cannot clearly represent and group together co-occurring events within the same trace across dimensions. *Second*, as such techniques often involve small and curated datasets, they have been proven several times as not scalable over real-world ones [11]. At the time of the writing, the most efficient temporal specification miner, Bolt2 [11], extracts data-less temporal clauses and, therefore, does not consider data payload information associated with each activity label, as required after extending DT with data payloads. As this was the one used in our previous work to characterize correlations across temporal dimensions after a minor extension for supporting durative and co-occurring events [16], the extraction of a declarative dataful specification requires to expand the latter algorithm further also to enable the extraction of dataful features from the declarative clauses.

This paper aims to bridge all approaches above to achieve a generally explainable sequential learning explanation through DataFul Explainable MultivariatE coRrelatIonal Temporal Artificial inTElligence (EMeriTAte+DF)[4]. The proposed technique works by first discretizing MTS into machine-readable polyadic logs through dataful DT mining (Sect. 4.1), for then extracting declarative descriptions of the previously-mined representation while pertaining dataful features (Sect. 4.2). We then use such clauses to derive the features satisfied, violated, or vacuously satisfied by the traces, for then deriving a propositional representation of the classes associated with arbitrary MTS segments through white-box classifier explanation extraction (also Sect. 4.2). By doing so, each temporal class is then defined through the satisfaction of a proportional formula, where predicates state temporal properties of the data. This approach further generalises pre-existing de-

---

[4] https://github.com/datagram-db/knobab/releases/tag/v3.1

viance mining solutions, which only encompass the conjunctive-based characterization of such temporal classes [14].

This constitutes a major improvement over our previous solution Explainable Multi-variatE coRrelatIonal Temporal Artificial inTElligence (EMeriTAte) [16], where *(i)* the original implementation of DT mining was not extracting a numerical summarised description of the data and *(ii)* was using a straightforward brute-force algorithm for considering all the possible sliding windows of all possible sizes; *(iii)* the extraction of the declarative clauses for each polyadic log (Sect. 3.1) did not consider dataful clauses, and *(iv)* the generation of the embedding from the mined clauses was regarded as a distinct phase from the one where the trace embeddings were generated as in previous literature [14], thus adding significant computational overhead. The application of the aforementioned changes to our previous solution enables us to design its dataful variant (EMeriTAte+DF) outperforming EMeriTAte in terms of efficiency, accuracy, precision, and recall. Last, we extend our experimental set-up to consider other MTS datasets, thus including univariate ones [26,21] and others related to general human mobility, thus remarking the inherent difficulty of characterizing dyskinetic/off events in Parkinson's Disease patients.

1.1. Notation

The following table provides the notation used in the rest of the paper.

Mathematical Notation	
$\langle \ldots \rangle$	sequence of index-ordered elements
$\{\ldots\}, \emptyset$	set (bag) of elements, empty set
$D, D_{\text{train}}, D_{\text{test}}$	dataset, train d., test d.
$\mathbb{N} = \{0, 1, \ldots\}$	set for natural numbers
$\mathbb{R}, \mathbb{R}^d$	sets for real numbers and d-dimensional real-valued vectors (resp.)
$|S| \in \mathbb{N}$	cardinality (or rank, size) of S
$f \colon A \to B$	function f with domain in A and codomain in B
$\tilde{p} \colon A \rightharpoonup B$	finite function $\tilde{p}$ with finite domain in A and codomain in B
$\mathrm{dom}(f) = A$	domain functor dom, it returns f's domain A
$\tilde{p} = [a \mapsto b, c \mapsto d]$	using the graph of a function to define a finite function $\tilde{p}$ with $\mathrm{dom}(\tilde{p}) = \{a, b\}$: $\tilde{p}(a) = b$ and $\tilde{p}(c) = d$
$[b, e] \simeq \langle b, b+1, \ldots, e \rangle$	numeric iterval, isomorphic to a sequence of natural numbers
$f \oplus g$	finite function concatenation [8]
$\mathbf{1}_b = \begin{cases} 1 & b = \textbf{true} \\ 0 & \text{oth.} \end{cases}$	indicator function
$\mathbb{B}(b) = \mathbf{1}_b - \mathbf{1}_{\neg b}$	A function mapping **true** boolean b values to 1 and **false** ones to -1.
$\mathbb{I}_F([B, E])$	set of maximal non-overlapping intervals in [B,E] sharing the same F values (Eq. 1)
Time Series	
$t, B, E, b, e, \beta, \eta$	(point) time indices (in $\mathbb{N}$)

$x \in \{\mathtt{i}, \mathtt{a}, \mathtt{s}, \mathtt{v}\}$	(numerical) condition index (Table 2)
$T \colon \mathbb{N} \rightharpoonup \mathbb{R}$	univariate Time Series (TS)
T^x	TS X satisfying a numerical condition x (string)
$T(t)$	TS value at time t
$T[\![B, E]\!] = \langle T(B), \ldots, T(E) \rangle$	univariate projection
$\mathfrak{E}, \tau \colon \mathbb{N} \rightharpoonup \mathbb{R}^d$	d-dimensional Multivariate Time Series (MTS) (or environment)
$\psi \in [1, d]$	MTS dimension index
$\mathfrak{E}_\psi$	TS representing MTS $\mathfrak{E}$ values for dimension ψ
$\mathfrak{E}_\psi^x$	MTS $\mathfrak{E}$ satisfying a numerical condition x over dimension ψ (string)
$\mathfrak{E}_\psi(t), \mathfrak{E}(t)(\psi)$	MTS value at time t and dimension ψ
$\mathfrak{E}_d(t) = \mathfrak{E}(t)(d) = \chi$	MTS being in class χ at time t ($d \geq 2$)
$\mathfrak{E}[B, E]$	MTS projection over a time interval $[B, E]$ (Eq. 2)
$val_x^{\mathfrak{E}}(t)$	numerical condition x value for MTS $\mathfrak{E}$ at time t (Table 2)
$P_x^{\mathfrak{E}}(t)$	point-wise predicate for MTS $\mathfrak{E}$ and condition x at time t (Table 2)

Polyadic Logs, DECLARE [36]

φ	DECLARE clause index (specification)
$i, j, k \in \mathbb{N}$	trace, event, and constituent index (resp.)
$\mathsf{A}, \mathsf{B}, T^{\mathtt{i}}, \mathsf{IncreaseRapidly}(T^{\mathtt{i}})$	activity labels (or strings)
$*$	always-matching activity label (or jolly, wildcard)
p, p', q, q'	data conditions
c_l	DECLARE template (string, Fig. 1b)
$c_l(\mathsf{A}, p)$	unary DECLARE clause
$c_l(\mathsf{A}, p, \mathsf{B}, q), \ c_l(\mathsf{A}, p, \mathsf{A'}, p')$	binary DECLARE clauses
$\Phi, \Phi_1, \Phi_2, \ldots$	set of specifications (with conjunctive interpretation)
$G_\psi = \langle N_{G_\psi}, R_{G_\psi}, \ell_{G_\psi} \rangle$	ℓ_{G_ψ}-rooted Direct Acyclic Graph with nodes in N_{G_ψ} and edges in R_{G_ψ} [12]
$\varsigma_{j,k}^i = \langle \mathsf{A}, \tilde{p}, \jmath \rangle$	k-th constituent for p. event σ_j^i, with an activity label A, a data payload $\tilde{p}$, and a duration span of $\jmath$
$\lambda(\varsigma_{j,k}^i) = \mathsf{A}$	activity label projection
$\varpi(\varsigma_{j,k}^i) = \tilde{p}$	payload projection
$\delta(\varsigma_{j,k}^i) = \jmath$	span projection
$\kappa(\varsigma_{j,k}^i)$	information concatenation (Eq. 3)
$\sigma_j^i = \langle \langle \varsigma_{j,1}^i, \ldots, \varsigma_{j,m}^i \rangle, \chi \rangle$	j-th poliadic event of p. trace σ^i with m constituents and class χ
$\sigma^i = \langle \sigma_1^i, \ldots, \sigma_o^i \rangle$	i-th polyadic trace, with $|\sigma^i| = o$
$\mathcal{L} = \langle \sigma^1, \ldots, \sigma^n \rangle$	sequence of n polyadic traces
$\mathfrak{S} = \langle \mathbb{G}, \mathcal{L} \rangle, \mathbb{G} = \{G_1, \ldots, G_n\}$	polyadic log (or system), taxonomies' set [35]
Σ	Alphabet, set of activity labels occurring in a dataset of logs (Algorithm 2, Line 2)

Algorithms	
$s\|t$	string concatenation[5], with implicit operands' casting
yield v	A function containing a **yield** statement is a generator function, returning all the v-s in a collection in order of generation[6].
INITFRESH(n)	Initializes all the auto-increment natural-valued variables ζ_i to n for all $i \in \mathbb{N}$
fresh(k)	Returns the current value of ζ_k before incrementing it. The term fresh comes from "fresh variable" [20]
$l \leftarrow r$	The reference to l is updated with the value r. If l does not exist, it gets declared[7].
DF	DataFrame [31]: a double-entry table providing cell value access v through row i and column j index, $DF[i,j] = v$
ZIP$(t_1, \dots) =$ $\dots, \langle t_1(i), t_2(i), \dots \rangle, \dots$	zip returns a sequence which i-th element is the sequence containing all the operands' i-th elements[8].
$\mathcal{G}_{x,b}^{\psi,\mathfrak{E}}$	Indexing $\mathfrak{E}$ for all maximal contiguous (time) intervals for which the values from dimension ψ always satisfy ($b = $ **true**) $P_x^{\mathfrak{E}}$.

2. Related Works

2.1. Hybridly Explainable Artificial Intelligence

The observations from which the utility of defining a hybrid explainable AI is deduced have their origins in the formalization of verifiable AI systems. At the basis of this framework [39], we always consider a model of the system to be verified ($\mathfrak{S}$), a model of the environment ($\mathfrak{E}$), and the formal property to be verified (φ), where all of these components might be expressed in logical form, so to provide correctness guarantees. Specification mining algorithms [11] extract φ from given system or environment of interest $\mathfrak{S} \vee \mathfrak{E}$, where deviance mining algorithms [14] are a specific case of the former, assuming *(i)* that environment states can be labelled and *(ii)* that we can extract one single specification per environment label, which provides the characterizing behaviour distinguishing a class from the others. A later survey [13] generalised the former into *hybridly explainable artificial intelligence* after observing that we can always transform unstructured data representations into more structured and logical-driven by pre-processing the data into an *a priori* phase, which can include both data cleaning capabilities [4] and specification mining ones by enriching structured data representation with contextual interpretations of the raw data. After this, we can carry out the actual learning phase, in which *ad hoc*

---

[5] https://datatracker.ietf.org/doc/html/rfc8017#section-2

[6] PHP: https://www.php.net/manual/en/language.generators.syntax.php, Python: https://docs.python.org/3/reference/expressions.html#yieldexpr

[7] https://www.overleaf.com/learn/latex/Algorithms

[8] Python: https://docs.python.org/3.3/library/functions.html#zip

explainability will heavily depend on the methodology and the data-representation format of choice. Last, we can consider precision/recall metrics as well as model extractions from both white-box and black-box classifiers as *ex post* explanations of the learning phase. This paper will focus on providing algorithmic solutions for the first two phases, as narrated in the text.

To better explain the former framework, we will categorize the forthcoming sections by pigeonholing curring literature over the aforementioned framework.

2.2.   A Priori Explanability

Numerical Data. An (univariate) TS $T\colon \mathbb{N} \rightharpoonup \mathbb{R}$ [22] of size $|T| = |\operatorname{dom}(T)|$ where $\operatorname{dom}(T) = \{1, \ldots, |T|\}$ is a finite function mapping time units $t \in \operatorname{dom}(T)$ into real values $T(t)$. Given a time interval $[b, e]$ and a TS X of at least size e, we denote $T[\![b, e]\!]$ as its *univariate projection* returning an ordered sequence $\langle T(b), T(b+1), \ldots, T(e) \rangle$. Thus, we consider intervals as a shorthand for an ordered sequence, $[b, e] \simeq \langle b, b+1, \ldots, e \rangle$. DT specifications (Fig. 1a) provide a LTL_f characterization for numerical increase $\mathtt{i}$ trends in (univariate) TS X after propositional discretization within a mining phase [24]: this requires a preliminary pre-processing step for which X is associated with durative constituents ς_{t+1}^T with activity labels $T^{\mathtt{i}}$ or $T^{\neg\mathtt{i}}$ depending on the value $T(t)$ of TS T at time t: ς_{t+1}^T has activity label $T^{\mathtt{i}}$ ($\lambda(\varsigma_{t+1}^T) = T^{\mathtt{i}}$) if $T(t+1) > T(t)$ and $T^{\neg\mathtt{i}}$ otherwise. We can then determine the occurrence of a DT pattern by simply joining constituents $T^{\mathtt{i}}$ and $T^{\neg\mathtt{i}}$ into durative ones referring to the same temporal span, which can then be matched and rewritten into a durative event having the activity label described in the first column of Fig. 1a. Authors mainly use this declarative representation for TS forecasting purposes, but give no evidence for exploiting this in the context of MTS while correlating disparate DT across MTS variables, which might be expressed through DECLARE (see **Event Data**). To achieve this, our previous work premiered their combination. Although DT discards the numerical information associated with such trends, it generalises the common shapelet approach [23] to arbitrary growth and decrease patterns expressed declaratively with human-understandable patterns.

Event Data. DECLARE [36] (Fig. 1b) is a temporal declarative language considering possible temporal behaviour in terms of correlations between activated events and targeted conditions, where the former refers to the necessary but not sufficient condition for satisfying a clause. Both activation and target conditions are defined through *activity labels* referring to a specific action of interest; these can be enriched with dataful propositional formulæ, predicating payload conditions over the activated constituents. Unless stated otherwise, if the clause has no activation, it is trivially satisfied (*vacuously*). E.g., Precedence might also be vacuously violated due to the presence of just target conditions [11]. DECLARE clauses can be instantiated over a specific alphabet Σ referring to activity labels associated with constituents and they can be composed to generate *conjunctive specifications* $\Phi = \{\varphi_1, \ldots, \varphi_n\}$; we say that a trace satisfies a conjunctive specification if it satisfies all its clauses. These conjunctive specifications can be extracted from temporal data represented as a (non-polyadic) log by efficiently exploiting a specific-to-general lattice search from which we can preventively prune the generation of specific clauses [11].

Activity Label	TS	Trace $(T^i, T^{\neg i})$
IncreaseRapidly(T^i)		$T^i, \ldots, T^i$
IncreaseSlowlyI(T^i)		$T^{\neg i}, T^i, \ldots, T^i$
IncreaseSlowlyII(T^i)		$T^i, T^{\neg i}, T^i, \ldots, T^i$
IncreaseSlowlyIII(T^i)		$T^i, \ldots, T^i, T^{\neg i}, T^i$
IncreaseSlowlyIV(T^i)		$T^i, \ldots, T^i, T^{\neg i}$
HighVolatilityI(T^i)		$T^{\neg i}, T^i$
HighVolatilityII(T^i)		$T^{\neg i}, T^i, T^{\neg i}, T^i$
HighVolatilityIII(T^i)		$T^{\neg i}, T^i, \ldots, T^i, T^{\neg i}$
HighVolatilityIV(T^i)		$T^i, T^{\neg i}, \ldots, T^{\neg i}, T^i$
HighVolatilityV(T^i)		$T^i, T^{\neg i}, T^i, T^{\neg i}$
HighVolatilityVI(T^i)		$T^i, T^{\neg i}$
DecreaseSlowlyI(T^i)		$T^{\neg i}, \ldots, T^{\neg i}, T^i$
DecreaseSlowlyII(T^i)		$T^{\neg i}, \ldots, T^{\neg i}, T^i, T^{\neg i}$
DecreaseSlowlyIII(T^i)		$T^{\neg i}, T^i, T^{\neg i}, \ldots, T^{\neg i}$
DecreaseSlowlyIV(T^i)		$T^i, T^{\neg i}, \ldots, T^{\neg i}$
DecreaseRapidly(T^i)		$T^{\neg i}, \ldots, T^{\neg i}$

(a) Expressing DT activity labels as TS silhouettes and pointwise constituents for the increase numerical condition, i [24].

Clauses φ from template c_l	Non-polyadic logs interpretation $[\![c]\!]$
†Init(A, p)	The trace should start with an activation
†End(A, p)	The trace should end with an activation
†Exists(A, p)	Activations should occur at least once
Absence(A, p)	Activations should never occur
Choice(A, p, A', p')	One of the two activation conditions must appear.
ExclChoice(A, p, A', p')	Only one type of activation is admitted per trace.
RespExistence(A, p, B, q)	An activation requires at least one target.
CoExistence(A, p, B, q)	$[\![\text{RespExistence}(A, \textbf{true}, B, \textbf{true})]\!] \wedge [\![\text{RespExistence}(B, \textbf{true}, A, \textbf{true})]\!]$
†Precedence(A, p, B, q)	Events preceding the activations should not satisfy the target.
†Response(A, p, B, q)	An activation leads to a future target event.
Succession(A, p, B, q)	$[\![\text{Precedence}(A, \textbf{true}, B, \textbf{true})]\!] \wedge [\![\text{Response}(A, \textbf{true}, B, \textbf{true})]\!]$
†ChainPrecedence(A, p, B, q)	An activation leads to an *immediately preceding* target.
†ChainResponse(A, p, B, q)	An activation leads to an *immediately following* target.
ChainSuccession(A, p, B, q)	$[\![\text{ChainPreced.}(B, \textbf{true}, A, \textbf{true})]\!] \wedge [\![\text{ChainResponse}(A, \textbf{true}, B, \textbf{true})]\!]$

(b) Our dataful DECLARE [36] subset of interest, where A (respectively, B) denote activation (resp., target), and p (resp., q) denote the associated dataful payload condition. conditions. Dataless variants can be expressed with p (p' and q) as **true**. Given an unary (respectively, binary) template c_l, $c_l(A, p)$ (resp., $c_l(A, p, B, q)$ or $c_l(A, p, A', p')$) denotes an unary (resp., binary) clause φ. †remarks clauses subject to dataful refinement in the present paper.

Fig. 1. Declarative Languages for *a priori* explanability over *(a)* TS or *(b)* non-polyadic logs.

Bolt2 [11] mines non-polyadic logs to extract conjunctive DECLARE specifications, where events are non-durative and where every single event contains one and only one constituent. The adoption of the KNOwledge Base for Alignments and Business process modelling (KnoBAB) [11] for storing the log and traces in primary memory allows fast constituent retrieval by activity label and trace scanning. This was possible as its primary goal was to maximize formal verification tasks over logs of traces. By offering indexed relational tables with temporal, payload, and event-counting data represented in a columnar way, we ensure fast data access for scanning multiple events occurring in a log. Bolt2 achieves efficient specification mining not only by the means of efficient data access via KnoBAB, but by also testing DECLARE dataless clauses where the most general clause is tested first, followed by the ones enabling the other. This generates a lattice of DECLARE clauses: the clause search stops by adopting a quality metric, thus ensuring the capture of the most specific behaviour of the single log without returning underrepresented ones as not generalizing over the data. This solution does not contemplate any further refinement of the clauses into dataful ones so to associate non-trivially true data predicates to either the activation, target, or correlation conditions. Other specification mining algorithms overcome this limitation [27] and, despite not using an efficient general-to-specific visit as the former, consider clustering over the activation conditions for deriving the data's propositional representations. This consideration was blindly adopted in Deviant Model Learning (DML) [14] to refine each mined declare clause to generate dataful clauses differentiating traces belonging to different classes. This poses a significant limitation in this scenario, as the mining approach was not initially designed for this purpose, thus not necessarily helping to refine the activation conditions according to the class associated with each single activation or target condition. This paper proposes a completely different approach, where white-box classifiers are used to characterize the activation (and target) predicates with the clear intent to capture distinct class behaviours and not merely aggregate data by shared data features, which might not be necessarily related to a class of interest.

2.3.   Ad Hoc Explanability

Numerical Data. *This subsection describes the main competitors considered from this paper for EMeriTAte+DF, as they constitute MTS classifiers.*

(k-nearest neighbors (KNN)-based) Clustering for time series [25] works by first identifying time series clusters depending on a distance function of choice and then associating to each cluster the majority class; this induces the possibility of boiling down a clustering problem to a classification one by associating each numerical-based representation to the element being the most similar to the majority class similarly to [7]. Euclidean k-nearest neighbors clustering (E-KNN) uses a straightforward Euclidean metric can determine the pointwise distance of the MTS across dimensions. The main drawback of this is that it neither considers the evolution of DTs within the MTS nor allows for aligning similar trends being displaced in time, similar to dynamic time warping. Furthermore, no mechanism efficiently tests different patterns at stake by considering a hierarchy of temporal patterns, one being the generalization of the other. Doing so can greatly boost the mining task while increasing the overall number of temporal patterns of interest.

Rocket [18] exploits randomized convolutional kernels to extract relevant features from the MTS, which are then fed to a linear classifier by associating such features to

a class of interest. While this feature extraction approach cannot adequately capture DTs and variations across dimensions, it also guarantees scarce explainability as information is summarised into kernel values. The Multivariate Time Series Classifier with Attentional Prototypical Network (TapNet) [43] improves the latter by exploiting attention networks for selecting the best classification features while still heavily relying on up-front feature selection mechanisms such as dimensionality reduction, which might lose relevant information to the detriment of the classification precision and accuracy. Canonical Interval Forest (CIF) [32] achieves explainability by exploiting white-box classifiers such as decision trees over catch22 features [30] describing a selection of time intervals describing different types of numerical variations rather than changes in trends and data patterns and their temporal correlations. Shapelet Transform Classifier (STC) [23] characterize MTS in terms of distinctive temporal features, the *shapelets*, being frequently occurring trends occurring across all TS, for then describing each MTS in terms of their distance with each selected shapelet. Notwithstanding this approach considers numerical features that are completely discarded through DT, it fails to establish correlations across trends occurring in different dimensions, thus not establishing correlations across differently observed behaviours.

Event Data. DML [14] extracts specifications for trace classes $y \in \mathcal{Y}$ through declarative (e.g., DECLARE clauses) and procedural (i.e., association rules) features previously mined from the data. Each trace can be represented as an n-dimensional embedding $\vec{x}_\sigma \in \mathbb{R}^n$, where n is the overall number of features. E.g., when a Declare clause φ is chosen as the ψ-th feature, a value of $\vec{x}_\sigma[\psi] = -1/0/n$ denotes that the trace σ does not satisfy, or vacuously satisfies, or satisfactorily activates φ n times. For the features providing declarative characterizations, DML extracts those via the ADMS specification mining algorithm [11]. Our previous approach, EMeriTAte [16], generalised over the former algorithm in its *ad hoc* phase by considering a different log model, where each event (and not just a single trace) is associated with a class and where each event is composed by different *constituents* representing concurrent activities starting at the same time and with different durations. These characteristics also enable the Event Data algorithms to consider numerical-driven problems with different dimensions. At the same time, previous temporal declarative-driven approaches were limited to only considering univariate TS [19], and were only used for verification tasks, but not for specification mining ones. We further expand on this data model to consider multiple taxonomies of activity labels for considering which events might be regarded as sub-events of one more general type (Sect. 3.1), so as to reduce better the amount of relevant correlations (Sect. 4.2).

3. Problem Statement

We define $\mathbb{B}(b) = \mathbf{1}_b - \mathbf{1}_b$ where $\mathbf{1}_b$ is the indicator function for predicates p. Given a function F having as a domain a closed interval in $\mathbb{N}$, the set $\mathbb{I}_F([B, E])$ of the maximal intervals in $[B, E] \subseteq \mathbb{N}$ is a set of largest non-overlapping intervals in $[B, E]$ sharing the same value for F where all the remaining intervals will have different values in F:

$$\mathbb{I}_F([B, E]) = \{[b, e] \mid B \leq b \leq e \leq E, (b > B \Rightarrow F(b) \neq F(B)),$$
$$(e < E \Rightarrow F(e) \neq F(E)), \forall b \leq t \leq e.F(t) = F(b) = F(e)\} \tag{1}$$

Given a d-dimensional MTS $\mathfrak{E} \colon \mathbb{N} \rightharpoonup \mathbb{R}^d$ where its last dimension d reflects the fluctuation of the class label in time, we refer to its *size* $|\mathfrak{E}|$ as the number of time (points) counting from 1, thus $\mathrm{dom}(\mathfrak{E}) = \{1, \dots, |\mathfrak{E}|\}$. We also denote $\mathfrak{E}(t)(\psi)$ as the value of the MTS $\mathfrak{E}$ at time point $t \in \mathrm{dom}(\mathfrak{E})$ in dimension $1 \leq \psi \leq d$. Given a time interval $[b, e]$ and a MTS $\mathfrak{E}$ of at least size e, we denote $\mathfrak{E}[b, e]$ as its *projection* considering a subsequence of the events occurring within this interval:

$$\mathfrak{E}[b, e](t) = \mathfrak{E}(t + b - 1) \textbf{ if } t + b - 1 \in \mathrm{dom}(\mathfrak{E}) \textbf{ and } 1 \leq t \leq e - b + 1 \tag{2}$$

Given a dimension $1 \leq \psi \leq d$, we denote $\mathfrak{E}_\psi$ as the TS providing $\mathfrak{E}$ values for dimension ψ: $\mathfrak{E}_\psi(t) = \mathfrak{E}(t)(\psi)$ for all $t \in \mathrm{dom}(\mathfrak{E})$.

Given a MTS $\mathfrak{E}$ within a training dataset D and a set of maximal and non-overlapping temporal subsequences $[b, e] \in \mathbb{I}_{\mathfrak{E}_d}([1, |\mathfrak{E}|])$ targeting class fluctuations in dimension d, we want to learn a function h minimising the classification error $\sum_{b \leq t \leq e} |h(\mathfrak{E}[b, e]) - \mathfrak{E}(t)(d)|$ for each maximal interval $[b, e]$ and from all $\mathfrak{E}$ in D. This is a more general formulation than the one posed by current literature, considering time series where $\mathbb{I}_{\mathfrak{E}_d}([1, |\mathfrak{E}|]) = \{[1, |\mathfrak{E}|]\}$, where one classification label is associated to each timestamp.

3.1.   Polyadic Logs

A *polyadic log* $\mathfrak{S}$ is a pair $\langle \mathbb{G}, \mathcal{L} \rangle$, where *(i)* $\mathcal{L}$ is a collection of distinct *polyadic traces* $\langle \sigma^1, \dots, \sigma^n \rangle$ referring to the auditing of a specific environment $\mathfrak{E}$ of interest (e.g., a MTS); each trace σ^i of length o is a list of temporally ordered *polyadic events* $\langle \sigma^i_1, \dots, \sigma^i_o \rangle$, where each of these σ^i_j of length m is defined as a pair of a set of *durative constituents* and a class $\chi \in \mathbb{R}$ $\sigma^i_j = \langle \langle \varsigma^i_{j,1}, \dots, \varsigma^i_{j,m} \rangle, \chi \rangle$, where all constituents start at the same time j but possibly with different activity labels and duration span; in particular, each durative constituent $\varsigma^i_{j,k}$ is expressed as a triplet $\langle \mathsf{a}, \tilde{p}, s \rangle$ where a is the activity label, $\tilde{p}$ is the payload collecting the raw data being associated to the durative constituent, and s denotes its temporal span s.t. $s \leq n - j + 1$. In this, we denote with λ, ϖ, and δ the function extracting the activity label ($\lambda(\varsigma^i_{j,k}) = \mathsf{a}$), its payload ($\varpi(\varsigma^i_{j,k}) = \tilde{p}$), and its span ($\delta(\varsigma^i_{j,k}) = s$). We denote $\kappa(\varsigma^i_{j,k})$ as the information concatenation function merging all of this information in the following finite function:

$$\varpi(\varsigma^i_{1,k}) \oplus [\_\_\mathrm{label} \mapsto \lambda(\varsigma^i_{1,k}), \_\_\mathrm{span} \mapsto \delta(\varsigma^i_{1,k})] \tag{3}$$

Last, *(ii)* $\mathbb{G}$ is a collection of *taxonomies* [35,12] represented as Direct (Acyclic) Graphs $G_\psi = \langle N_{G_\psi}, R_{G_\psi}, \ell_{G_\psi} \rangle$ rooted in ℓ_{G_ψ} determining the relationship between activity labels within the log $\mathfrak{S}$; the entities are represented as a set of nodes N_{G_ψ}, and R_{G_ψ} is the set of the is-a relationships.

3.2. Poly-DECLARE

Polyadic traces trivially violate the temporal non-simultaneity assumption from non-polyadic logs [10] for which each non-polyadic event cannot be associated with multiple activity labels, while now one single polyadic event might contain constituents with different activity labels. As a result, the former algorithms did not consider concurrent violation conditions: e.g., for Precedence(A,**true**,C, **true**), we now also prescribe that a C-labelled constituent shall also never co-occur with an A-labelled constituent (cfr. the occurrence of $\leq$, Algorithm 4). Achieving all these desiderata requires completely revising the previous algorithms [11,14] to be adapted to the novel log assumptions (see Algorithm 5).

Given that each event might contain multiple constituents, we extend the traditional DECLARE semantics by checking whether some or all the constituents activated within a specific event satisfy the given declarative clause. We retain the former as the default interpretation of the declared clauses, thus retaining the same template name while, for indicating the others, we explicitly append a All prefix to the templates c_l from Fig. 1b as $\text{All}\|c_l$.

4.   EMeriTAte+DF

This section introduces the algorithmic extensions moving from EMeriTAte to its DATA-FUL *(DF) variant (EMeriTAte+DF). Unlike our previous contribution, we characterize our algorithms as Explainable and Verified AI. While the A Priori phase adds more contextual information on the data series, the Specification Mining and the White-Box model learning for learning the explainable specifications from the data is carried out in the Ad Hoc phase. The evaluation section provides the Post Hoc evaluation of the trained models under different datasets using the default metrics.*

4.1.   A Priori Explainability

The a priori explainability phase was improved from [24] as Algorithm 1 by adequately indexing MTS to support faster DT mining algorithms while also simplifying over the redundant patterns from Fig. 1a. Differently from the previous, we extend each constituent having DT activity labels, once represented as a dataless durative constituent, with catch22 features as data payloads for the mined durative constituents.

Algorithm 1 A Priori Explainability

1: **procedure** TSLOADINDEX($\mathfrak{E}; \varepsilon$)
2:   **for all** $[B, E] \in \mathbb{I}_{\mathfrak{E}_d}([1, |\mathfrak{E}|])$ **do**
3:     $\tau \leftarrow \mathfrak{E}[B, E];\quad \chi \leftarrow \tau(1)(d)$
4:     **for** ψ **from** 1 to $d - 1$ **do**
5:       **for all** $x \in \{\mathtt{i}, \mathtt{a}, \mathtt{s}, \mathtt{v}\}$ **do**
6:         $\mathcal{L}_x^{\psi, \tau} \leftarrow \langle \sigma_1^{\psi, \tau}, \dots, \sigma_{E-B+1}^{\psi, \tau} \rangle$ $\triangleright$ *Loading*
7:         **for all** $1 \leq t \leq E - B + 1$ **do**
8:           $\varsigma_{t,x}^{\psi, \tau} \leftarrow \langle P_x^\tau(t), [\psi \| x \mapsto val_x^\tau(t)], 1 \rangle$
9:           $\sigma_t^{\psi, \tau} \leftarrow \langle \langle \varsigma_{t,x}^{\psi, \tau} \rangle, \chi \rangle$
10:        $\mathcal{G}_{x,\mathbf{true}}^{\psi, \tau} \leftarrow \big\{ [b, e] \in \mathbb{I}_{P_x^\tau}([B, E]) \mid \forall b \leq t \leq e.\, P_x^\tau(t) \big\} \triangleright$*Indexing, P_x^τ via $\mathcal{L}_x^{\psi, \tau}$*
11:        $\mathcal{G}_{x,\mathbf{false}}^{\psi, \tau} \leftarrow \big\{ [b, e] \in \mathbb{I}_{P_x^\tau}([B, E]) \mid \forall b \leq t \leq e.\, \neg P_x^\tau(t) \big\}$

Loading and Indexing. The first step is to *load* the MTS discretized as a preliminary pointwise log of events expressing the verification or not of a specific *numerical condition* x expressing different types of numeric variations (Table 2) such as increase ($\mathtt{i}$), absence ($\mathtt{a}$), stationarity ($\mathtt{s}$), and variability ($\mathtt{v}$), and by creating pointwise events satisfying or not some associated condition $P_x^\tau(t)$. This will then be used to mine the DT patterns.

Table 2. Point-wise predicates P_x^τ and values val_x^τ for discretizing time series τ over dimension i and numerical conditions x

x	$val_x^\tau(t)$	$P_x^\tau(t)$
$\mathtt{i}$	$\tau(t+1)(i) - \tau(t)(i)$	$val_{\mathtt{i}}^\tau(t) > 0$
$\mathtt{a}$	$\|\tau(t)(i)\|$	$val_{\mathtt{i}}^\tau(t) \leq \varepsilon$
$\mathtt{s}$	$\|val_{\mathtt{i}}(t)\|$	$val_{\mathtt{s}}^\tau(t) \leq \varepsilon$
$\mathtt{v}$	$\begin{cases} 0 & P_{\mathtt{s}}^\tau(t) \\ +\infty & P_{\mathtt{a}}^\tau(t) \\ \frac{val_{\mathtt{i}}^\tau(t)}{\tau(t)(d)} & \text{oth.} \end{cases}$	$P_{\mathtt{s}}^\tau(t)$

We perform a linear scan of each multivariate time series $\mathfrak{E}$ so to identify the maximal intervals $[B, E] \in \mathbb{I}_{\mathfrak{E}_d}([1, |\mathfrak{E}|])$ associated with the same class reported in dimension d (Line 2), thus considering a projection τ for each of these $[B, E]$. For each dimension of interest (Line 4), we discretize each dimension and timestamp as in our previous paper [16] by considering numerical variations (Line 5). All such constituents are then stored in events (Line 9) contained in $\mathcal{L}_x^{\psi, \tau}$ for each dimension ψ, segment $[B, E]$ as τ, and numerical variation x (Line 6).

Next, we *index* the previously-discretized logs according to the satisfaction (Line 10) or violation (Line 11) of the predicate P_x^τ. We ensure that this information is computed only once by identifying regions of a specific time series' dimension where numerical

12: **function** $\mathrm{DL}(\ell, \nu; \mathfrak{E}_\psi^x)$ ▷ *Generating DT activity labels*

13:     **if** $\ell = \mathrm{S}$ **then return** (ν **?** IncreaseRapidly **:** DecreaseRapidly) $\|(\mathfrak{E}_\psi^x)$

14:     **if** $\ell = \mathrm{HV43}$ **then return** (ν **?** HighVolatilityIII **:** HighVolatilityIV) $\|(\mathfrak{E}_\psi^x)$

15:     **if** $\ell = \mathrm{1H}$ **then return** (ν **?** IncreaseSlowlyI **:** DecreaseSlowlyIV) $\|(\mathfrak{E}_\psi^x)$

16:     **if** $\ell = \mathrm{2H}$ **then return** (ν **?** IncreaseSlowlyII **:** DecreaseSlowlyIII) $\|(\mathfrak{E}_\psi^x)$

17:     **if** $\ell = \mathrm{E1}$ **then return** (ν **?** IncreaseSlowlyIII **:** DecreaseSlowlyII) $\|(\mathfrak{E}_\psi^x)$

18:     **if** $\ell = \mathrm{E2}$ **then return** (ν **?** IncreaseSlowlyIV **:** DecreaseSlowlyI) $\|(\mathfrak{E}_\psi^x)$

19: **function** $\mathrm{C22SHORT}(\mathfrak{E}_\psi, b, e)$ ▷ *Combining the TS patterns with the time ones*

20:     **return** $\mathrm{C22}(\mathfrak{E}_\psi[\![b,e]\!]) \oplus \mathrm{C22}([\mathrm{b,e}])$

21: **procedure** $\mathrm{DTMINESTEP}(\tau, \psi, x, \nu, [b,e]; \mathfrak{E})$

22:     $n \leftarrow e - b + 1$

23:     **yield** $\varsigma_{b,\mathrm{fresh}(b)}^{\mathfrak{E}} \leftarrow \langle \mathrm{DL}(\mathrm{S}, \nu; \mathfrak{E}_\psi^x), \mathrm{C22SHORT}(\mathfrak{E}_\psi, b, e), n \rangle$

24:     **for all** $1 \leq t \leq n$ **do**   ▷ *Determining the span $\jmath$ from t*

25:        **for all** $b \leq \beta \leq e - t + 1$ **do**

26:           $\xi \leftarrow (e+1) \in \mathcal{G}_{x,\mathbf{not}\,\nu}^{\psi,\tau}$ **and** $\beta = e$; $\eta \leftarrow \beta + t - 1$

27:           **if** $\beta = b$ **then**

28:              **if** $(b-1) \in \mathcal{G}_{x,\mathbf{not}\,\nu}^{\psi,\tau}$ **then**

29:                 **yield** $\varsigma_{\beta-1,\mathrm{fresh}(\beta-1)}^{\mathfrak{E}} \leftarrow \langle \mathrm{DL}(\mathrm{1H}, \nu; \mathfrak{E}_\psi^x), \mathrm{C22SHORT}(\mathfrak{E}_\psi, \beta-1, \eta),$

30:                                  $t+1 \rangle$

31:              **if** ξ **then**

32:                 **yield** $\varsigma_{\beta-1,\mathrm{fresh}(\beta-1)}^{\mathfrak{E}} \leftarrow \langle \mathrm{DL}(\mathrm{HV43}, \nu; \mathfrak{E}_\psi^x),$

33:                          $\mathrm{C22SHORT}(\mathfrak{E}_\psi, \beta-1, \eta+1), t+2 \rangle$

34:              **if** $(\beta-2) \in \mathcal{G}_{x,\nu}^{\psi,\tau}$ **then**

35:                 **yield** $\varsigma_{\beta-2,\mathrm{fresh}(\beta-2)}^{\mathfrak{E}} \leftarrow \langle \mathrm{DL}(\mathrm{2H}, \nu; \mathfrak{E}_\psi^x), \mathrm{C22SHORT}(\mathfrak{E}_\psi, \beta-2, \eta),$

36:                                  $t+2 \rangle$

37:           **if** $\beta + t - 1 = e$ **and** ξ **then**

38:              found $\leftarrow$ **false**

39:              **if** $[e+1, e+2] \in \mathcal{G}_{x,\mathbf{not}\,\nu}^{\psi,\tau}$ **and** $e+2 \in \mathcal{G}_{x,\nu}^{\psi,\tau}$ **then**

40:                 found $\leftarrow$ **true**

41:                 **yield** $\varsigma_{\beta,\mathrm{fresh}(\beta)}^{\mathfrak{E}} \leftarrow \langle \mathrm{DL}(\mathrm{E1}, \nu; \mathfrak{E}_\psi^x), \mathrm{C22SHORT}(\mathfrak{E}_\psi, \beta, \eta+2), t+2 \rangle$

42:              **if not** *found* **then**

43:                 **yield** $\varsigma_{\beta,\mathrm{fresh}(\beta)}^{\mathfrak{E}} \leftarrow \langle \mathrm{DL}(\mathrm{E2}, \nu; \mathfrak{E}_\psi^x), \mathrm{C22SHORT}(\mathfrak{E}_\psi, \beta, \eta+1), t+1 \rangle$

44: **procedure** $\mathrm{DTMINE}(\mathfrak{E})$

45:     $S^{\mathfrak{E}} \leftarrow \emptyset$; $\sigma^{\mathfrak{E}} \leftarrow \langle \rangle$; $\mathrm{INITFRESH}()$

46:     **for all** $[B, E] \in \mathbb{I}_{\mathfrak{E}_d}([1, |\mathfrak{E}|])$ **do**

47:        $\tau \leftarrow \mathfrak{E}[B, E]$; $\chi \leftarrow \tau(1)(d)$

48:        **for all** $1 \leq t \leq E - B + 1$ **do**

49:           $\varsigma_{B+t-1,1}^{\mathfrak{E}} \leftarrow \langle \_\_\mathsf{raw\_data}, [\mathrm{dim\_}\psi \mapsto \tau(t)(\psi)]_{1 \leq \psi \leq d-1}, 1 \rangle$

50:           $S_{B+t-1}^{\mathfrak{E}} \leftarrow [\varsigma_{B+t-1,1}^{\mathfrak{E}}]$; $\sigma_{B+t-1}^{\mathfrak{E}} \leftarrow \langle S_{B+t-1}^{\mathfrak{E}}, \chi \rangle$; $\sigma^{\mathfrak{E}} \leftarrow \sigma^{\mathfrak{E}} \cup [\sigma_{B+t-1}^{\mathfrak{E}}]$

51:        **for** ψ **from** 1 **to** $d - 1$ **do**

52:           **for all** $x \in \{\mathrm{i}, \mathrm{a}, \mathrm{s}, \mathrm{v}\}$ **do**

53:              **for all** $[b, e] \in \mathcal{G}_{x,\nu}^{\psi,\tau}$ **do**

54:                 $\varpi(\varsigma_{B-b+1,1}^{\mathfrak{E}}) \leftarrow \varpi(\varsigma_{B-b+1,1}^{\mathfrak{E}}) \oplus \varpi(\varsigma_{B-b+1,x}^{\psi,\tau})$

55:                 $S^{\mathfrak{E}} \leftarrow S^{\mathfrak{E}} \cup \mathrm{DTMINESTEP}(\tau, \psi, x, \mathbf{true}, [b, e]; \mathfrak{E})$

56:                 $S^{\mathfrak{E}} \leftarrow S^{\mathfrak{E}} \cup \mathrm{DTMINESTEP}(\tau, \psi, x, \mathbf{false}, [b, e]; \mathfrak{E})$

57:     **for all** $\varsigma_{j,k}^{\mathfrak{E}} \in S^{\mathfrak{E}}$ **do** $S_j^{\mathfrak{E}} \leftarrow S_j^{\mathfrak{E}} \cup [\varsigma_{j,k}^{\mathfrak{E}}]$

58:     **return** $\sigma^{\mathfrak{E}}$

variation conditions continue to hold or not. As the intervals in such indices are never overlapping, we can store the intervals in order of increasing beginning time and query those in $O(\log |\mathcal{G}_{x,\_}^{\psi,\tau}|)$ to check whether the time series at dimension i satisfies the numerical variation condition x at a specific time t. Hence, we denote and define such query as $t \in \mathcal{G}_{x,\_}^{\psi,\tau} \Leftrightarrow \exists [b,e] \in \mathcal{G}_{x,\_}^{\psi,\tau}. \, t \in [b,e]$: this notion is used in the forthcoming subsection.

Lemma 1 (Indexing and Loading Time Complexity). *Given a collection of N MTS of d dimensions with maximum length t, the time complexity for* TSLOADINDEX *is in* $O(Ndt)$.

Proof. Given that all insertions on indices can be built while linearly scanning the data, and assuming that each $P_x^\tau(x)$ can be tested in constant time, the time complexity of scanning each maximal time interval $[B, E]$ and each value in it boils down to the maximal time series length t. Given that the number of point-wise predicates is constant, we can conclude that the linear scan is provided for each dimension and MTS within the collection, thus leading to $O(Ndt)$. □

Indexed DT Mining. This phase enables the representation of a time series as a polyadic trace, where the first constituent for each j-th event reflects the values associated with each dimension at time j, i.e. $\mathfrak{E}(j)$ (Lines 49-50), while the remainder is derived from the DT-mined pointwise constituents from the previous phase (Lines 55-56). The composition of the $\mathcal{G}$-indexed intervals generate DT patterns distinguishable from a constituent activity label (Line 12), where the name of the DT pattern reports the dimension ψ and the specific numerical variation x. The algorithm proceeds by scanning each maximal interval $[B, E]$ for a time series $\mathfrak{E}$ over each dimension ψ, numerical variation x, and $\mathcal{G}$-indexed satisfiability interval $[b, e]$ (Lines 55-56). The segmentation of the time series through the $\mathcal{G}$ indices allows to promptly derive that any subsequent or preceding interval within the same dimension will violate some x constraints if the current interval satisfies them, and vice versa. Differently from our previous paper, for each mined constituent referring to a specific dimension i and interval $[\beta, \eta] \subseteq [b, e]$, we generate a payload containing the univariate catch22 [30] features (C22) capturing the dynamic properties of the ψ-th dimension in $\mathfrak{E}_\psi[\beta, \eta]$, which are paired with the time fluctuations (Line 20). By doing so, we generalise over the CIF classifier [32], which focuses on distinguishing time series from the behaviours occurring within a specific fixed-size sliding window while potentially considering the features for all possible sliding windows captured as DT patterns. The computation of the Catch22-payload is parallelized due to the high computational nature of the metrics used to describe the numerical fluctuations of the data. The main mining algorithm considers a subset of all the DTs patterns described in Section 2.2, where some high volatility patterns are discarded so to favour the recognition of shorter volatility patterns rather than preferring joining prolonged variations. Overall, this helps to reduce the number of patterns to be returned through the combination of subsequent patterns while reducing the time to mine the clauses without considerably undermining the predictive power of the classifier. At the end of the process, each time series $\mathfrak{E}$ is discretized into its constituents and collected in a single polyadic trace $\sigma^{\mathfrak{E}}$.

Lemma 2. *The worst-case time complexity for the indexed DT mining from* DTMINE *is superpolynomial over the maximum length of the segmented interval δ and linear over the number and size of the MTS.*

Proof. Note that the worst-case scenario time complexity does not happen for MTS exhibiting stable values, as this would lead to just one maximal interval $[B, E]$ per class. In fact, these situations have ξ and the condition at Line 28 as false, as there will be no other neighbouring intervals to consider. This will only lead to returning the durative constituent at Line 23, where still catch22 statistics are computed. As the time complexity of computing all the catch22 statistics depends on the length of the interval of interest, we denote this as $C_{\jmath}$.

Thus, the worst time complexity happens when we have at least two intervals, leading not only to the generation of the aforementioned durative constituent, but potentially to other patterns occurring within the nested iteration. Given that within the nested loop leading to $O(\jmath^2)$ iterations we test queries having a time in $O(\log \jmath)$, we obtain an overall time complexity of $O(\jmath^2 \log \jmath C_{\jmath} \frac{t}{\jmath} N)$. $\qquad\square$

Serialization At serialization time, we avoid writing any potential redundant constituent having the same activation label but referring to a shorter time interval to the other sibling constituents, thus capturing the behaviour associated with the longest span. We generate a collection of taxonomies for reconducing each constituent label to the original MTS dimension the pattern was referring to: we generate a taxonomy $G_{\psi} = \langle N_{G_{\psi}}, R_{G_{\psi}}, \ell_{G_{\psi}} \rangle$ for each dimension ψ, where $N_{G_{\psi}}$ contains all the DTs associated with $\mathfrak{E}_{\psi}^x$ thus including $\mathfrak{E}_{\psi}^x$ acting as the root $\ell_{G_{\psi}}$; $R_{G_{\psi}}$ connects only the root with the associated DT labels. Given the impossibility of serializing such data in XES [2] due to their strict assumptions on the log representation, we serialize the resulting information on a customary representation in `json`.

4.2.   Ad Hoc Explainability

We load the single log composed of many traces with classes changing in time as many segmented logs $\mathfrak{S}_y$ as the total number of classes $y \in \mathcal{Y}$ distinctly occurring by segmenting the traces in maximal intervals where the same class holds. Then, differently from our previous solution, where each segmented log was mined separately via Bolt2 [11] to then derive the trace embedding in a later phase, we contemporarily load and mine all the segmented logs containing maximal sequences referring to the same classes. We then proceed to the DECLARE clause refinement for the ones shared across the logs as a result of the mining phase: as generic dataless DECLARE clauses with DT activity labels might be not sufficient to temporally characterize the classes, we try to specialize each clause according to the specificity of each log/class. The rationale is the following: if the same dataless clause is frequently mined from two segmented logs, at least one trace exists from each of these where the dataless clause is both activated and holds. Then, the only way to differentiate the clause across the two segmented logs is to change the satisfiability outcome of the trace by refining the original dataless clause with dataful predicates for either the activation or the target condition. Suppose the payload referring to the constituents activated by the clause can be differentiated. In that case, we can refine the original dataless clause as two dataful ones, each with an activation condition (p and p') satisfying only the activations from one segmented log. This leads to one dataful clause being then activated over the traces of one segmented log, but being vacuously satisfied by the traces of the other. If that will not suffice, then the only further possible way to differentiate is to

generate target predicates similar to those above: in this other scenario, the missed sat-isfaction of the target condition will lead to a violation of the clause, thus further refining the trace behaviour in terms of clause (un)satisfaction.

This significantly differs from other deviance mining approaches [27], where acti-vation conditions are mainly clustered not necessarily according to the target classes of interest [14], thus not achieving a truthful clause separation for differentiating them across the distinct logs as discussed above. By exploiting a decision tree as a classifier, we can ensure that the data predicates used to differentiate the classes will summarize the data properties as unary dataful predicates, which might reconstructed by traversing each branch of the learned decision tree by interpreting it as a single propositional formula π.

Trace Loading and Segmentation. Similarly to our previous paper, for each trace $\sigma^{\mathfrak{E}_A}$ in the `json`-serialized log $\mathfrak{S}$ describing a distinct MTS $\mathfrak{E}$, we scan all polyadic events, and we group them by maximal contiguous sequences associated to the same class y. Each class y will then correspond to one single final log $\mathfrak{S}_y$, where each maximal contiguous sequence of polyadic events will correspond to one single new polyadic trace in $\mathfrak{S}_y$. As this transformation also preserves the __raw_data information associated with each polyadic event, we later collect the raw data associated with the polyadic traces for later on running atemporal event classification tasks by just correlating the class within the __raw_data payload with the other available values. As polyadic events now supports multiple constituents, this required us to update our KnoBAB solution to both extend the events' tabular representation (*activity table*), as well as extending the secondary in-dex structures for accessing multiple immediately preceeding or subsequent constituents within the same trace.

Polyadic Deviant Model Learning. Algorithm 2 showcases the procedure for extracting an explainable specification capturing MTS classes' temporal characterization by setting χ as the `class` associated with the i-th trace from $\mathfrak{S}_\chi$ (Line 7). This is achieved by considering DECLARE clauses, be they data or dataless, as features for describing class-segmented traces. By interpreting the numerical values as a violation, satisfaction, or vac-uous satisfaction of a specific clause, we then use a white box classifier such as Decision Tree to extract a propositional representation for the clauses' satisfaction.

Similarly to our previous implementation [11,16], we consider all the most frequent itemsets from FREQUENTITEMSETS for at most two activity labels and support of at least θ (Line 8). Unary DECLARE clauses such as Init, End, and Exists are mined from the frequently-occurring unary patterns in Φ_χ as per Bolt2 (Line 9). We further proceed on the refinement for each of the clauses into their dataful counterpart if and only if the payload-based refinement allows us to substantially differ the traces' behaviour according to the constituents' payloads and, otherwise, we backtrack to the original dataless variants (Line 21). Similar considerations can be carried out for the remaining binary clauses in FreqPairs$_\chi$: given all the possible binary frequent patterns occurring across all logs (Line 12), we perform no refinement if such pair is frequent only in one segmented log (Line 14) and, otherwise, we proceed by refining the clauses when possible (Line 17). Last, we train the white-box classifier over the extracted embedding for each class-segmented trace stored as a DF row, from which we derive a propositional [13] and declare-based [14] characterization for each MTS class (Line 22).

Algorithm 2 Ad Hoc Explainability

1: **function** POLYADICDATAAWARE($\mathfrak{S}_1, \ldots, \mathfrak{S}_n; \theta$)
2: $\quad \Sigma \leftarrow \bigcup_{\mathfrak{S}_\chi = \langle \mathbb{G}_\chi, \mathcal{L}_\chi \rangle, \langle N_{G_\psi}. R_{G_\psi}, \ell_{G_\psi} \rangle \in \mathbb{G}_\chi} N_{G_\psi}$ $\triangleright$ *Determining the alphabet*
3: $\quad \Phi \leftarrow \emptyset$ $\triangleright$ *Clauses not requiring dataful refinement*
4: $\quad$ **global** DF $\triangleright$ *Double-entry table initialization*

5: $\quad$ **for all** $1 \leq \chi \leq n$ **do**
6: $\quad\quad$ **for all** $\sigma^i \in \mathcal{L}, \mathfrak{S}_\chi = \langle \mathbb{G}_\chi, \mathcal{L}_\chi \rangle$ **do**
7: $\quad\quad\quad$ DF$[(\chi, i), \texttt{class}] \leftarrow \chi$ $\triangleright$ *Segmented trace/class assoc.*
8: $\quad\quad\quad$ $S_\chi \leftarrow$ FREQUENTITEMSETS$(\mathfrak{S}_\chi, \theta)$ $\triangleright$ *Maximal length of 2*
9: $\quad\quad\quad$ FreqPairs$_\chi, \Phi_\chi \leftarrow$ GENERATEUNARYCLAUSES(S_χ) $\triangleright$ *Unary clauses* [11]
10: $\quad$ $\Phi \leftarrow$ UNARYREFINE$(\Phi_1, \ldots, \Phi_n, \Phi; \mathfrak{S}_1, \ldots, \mathfrak{S}_n, \Sigma)$ $\triangleright$ Algorithm 3

11: $\quad$ FP $\leftarrow \bigcup_{1 \leq \chi \leq n}$ FreqPairs$_\chi$
12: $\quad$ **for all** $\langle \mathsf{A}, \mathsf{B} \rangle \in$ FP **do**
13: $\quad\quad$ $\Phi_{\mathsf{AB}} \leftarrow \{ c_l(\mathsf{A}, \mathbf{true}, \mathsf{B}, \mathbf{true}) | c_l \in BinaryTemplates \}$ $\triangleright$ *Probing all clauses*
14: $\quad\quad$ **if** $\exists! j. \langle \mathsf{A}, \mathsf{B} \rangle \in$ FreqPairs$_j$ **then**
15: $\quad\quad\quad$ $\Phi \leftarrow \Phi \cup \Phi_{\mathsf{AB}}$
16: $\quad\quad$ **else**
17: $\quad\quad\quad$ $\Phi \leftarrow \Phi \cup$ BINARYREFINE$(\mathsf{A}, \mathsf{B}, \mathfrak{S}_1, \ldots, \mathfrak{S}_n; \mathbf{true})$ $\triangleright$ Algorithm 4
18: $\quad$ **for all** $1 \leq \chi \leq n$ **do** $\triangleright$ *Dataless [16]*
19: $\quad\quad$ **for all** $\sigma^i \in \mathcal{L}, \mathfrak{S}_\chi = \langle \mathbb{G}_\chi, \mathcal{L}_\chi \rangle$ **do**
20: $\quad\quad\quad$ **for all** $\varphi \in \Phi$ **do**
21: $\quad\quad\quad\quad$ DF$[(\chi, i), \varphi] \leftarrow \mathbb{B}(\sigma^i \models \varphi)$
22: $\quad$ **return** DECISIONTREE$\Big(\Big\{ ([k \mapsto \mathrm{DF}[r, k]]_{k \in \texttt{cols}(\mathrm{DF}) \setminus \{\texttt{class}\}}, \mathrm{DF}[r, \texttt{class}])$
23: $\quad\quad\quad\quad\quad\quad\quad\quad\quad \Big| r \in \texttt{rows}(\mathrm{DF}) \Big\} \Big)$

Unary Refinement. Algorithm 3 outlines the dataful refinement of the unary clauses of (All)Init, (All)End, and (All)Exists, which is attempted if and only if we can mine the same clause from at least two class-segmented logs (Lines 17, 23, and 29). When this occurs, we collect in D all the payloads associated with the constituents occurring at the beginning (Line 20), end (Line 26) or from any constituent (Line 32) according to the clause to be refined. Lines 9 and 10 achieve the polyadic extension for DECLARE by considering the All$\|c_l$ clause variaties: this is achieved by not considering one payload at a time, but by considering the payload of all the constituents of interest cotemporarily, as stored in $\mathscr{D}$. We also consider all the predicates given within the decision tree and representing one class of interest and put them in disjunction, thus considering conditions generally identifying classes (Line 11). No refinement is provided if the decision tree cannot adequately separate the activation payloads according to the associated class to generate refined dataful activation predicates (Line 6).

As the explanation generated from the decision tree's *ex post* phase provides a propositional formula of declarative clauses, we discard the refinement for Absence, as this can still appear as the negation of any Exist clause [10].

Binary Refinement. Algorithm 4 describes refining some Polyadic DECLARE binary clauses. Given that the clauses expressing (Excl)Choice, CoExistence, and RespExis-

---

Algorithm 3 Ad Hoc Explainability: unary clause dataful refinement

---

1: **function** REFINEATTEMPT($D, \mathcal{D}, c_l, n; \vartheta = .7$)
2:    **global** DF
3:    $D_{\text{train}}, D_{\text{test}} \leftarrow$ SPLIT(D, ϑ)
4:    $\mathcal{M} \leftarrow$ DECISIONTREE(D_{train})
5:    $\mathcal{Y} \leftarrow [\chi \mapsto \bigwedge_{\pi \in m, \pi = H \Rightarrow \chi} H]_{1 \leq \chi \leq n}$
6:    **if** ACCURACY($\mathcal{M}, D_{\text{test}}$)$\leq 50\%$ **then return false**
7:    **for all** $(\chi, \sigma^i) \in \text{dom}(\mathcal{D})$ **do**
8:      **for all** $\pi \in \mathcal{M}$ **do**
9:        DF$[(\chi, i), \text{All}\|c_l(*, \pi)] \leftarrow \mathbb{B}(|\mathcal{D}(\chi, \sigma^i)| = \sum_{p \in \mathcal{D}(\chi, \sigma^i)} \mathbf{1}_{\pi(p)})$
10:        DF$[(\chi, i), c_l(*, \pi)] \leftarrow \mathbb{B}(0 < \sum_{p \in \mathcal{D}(\chi, \sigma^i)} \mathbf{1}_{\pi(p)})$
11:      **for all** $\chi \in \text{dom}(\mathcal{Y})$ **do**
12:        $\pi \leftarrow \mathcal{Y}(\chi)$
13:        DF$[(\chi, i), \text{All}\|c_l(*, \pi)] \leftarrow \mathbb{B}(|\mathcal{D}(\chi, \sigma^i)| = \sum_{p \in \mathcal{D}(\chi, \sigma^i)} \mathbf{1}_{\pi(p)})$
14:        DF$[(\chi, i), c_l(*, \pi)] \leftarrow \mathbb{B}(0 < \sum_{p \in \mathcal{D}(\chi, \sigma^i)} \mathbf{1}_{\pi(p)})$
15:    **return true**

16: **function** UNARYREFINE($\Phi_1, \ldots, \Phi_n, \Phi; \mathfrak{S}_1, \ldots, \mathfrak{S}_n, \Sigma$)
17:    refineInit $\leftarrow \exists \mathsf{a} \in \Sigma . \exists i, j \in \mathbb{N} . \mathsf{Init}(\mathsf{a}, \mathbf{true}) \in \Phi_i \cup \Phi_j$
18:    **if** refineInit **then** $\triangleright$ Init *refinement*
19:      $\mathcal{D} \leftarrow \left[(\chi, \sigma^i) \mapsto S\right]_{1 \leq \chi \leq n, \mathfrak{S}_\chi = \langle \mathbb{G}, \mathcal{L} \rangle, \sigma^i \in \mathcal{L}, \sigma_1^i = \langle S, \chi \rangle}$
20:      $D \leftarrow \left\{ (\kappa(\varsigma_{1,k}^i), \chi) \mid \varsigma_{1,k}^i \in \sigma_1^i, \sigma^i \in \mathcal{L}, \mathfrak{S}_\chi = \langle \mathbb{G}, \mathcal{L} \rangle, 1 \leq \chi \leq n \right\}$
21:      refineInit $\leftarrow$ REFINEATTEMPT($D, \mathcal{D}, \mathsf{Init}, n$)
22:    **if not** refineInit **then** $\Phi \leftarrow \Phi \cup \{\mathsf{Init}(\mathsf{b}, \mathbf{true}) | \mathsf{Init}(\mathsf{b}, \mathbf{true}) \in \bigcup_{1 \leq \chi \leq n} \Phi_\chi\}$

23:    refineEnd $\leftarrow \exists \mathsf{a} \in \Sigma . \exists i, j \in \mathbb{N} . \mathsf{End}(\mathsf{a}, \mathbf{true}) \in \Phi_i \cup \Phi_j$
24:    **if** refineEnd **then** $\triangleright$ End *refinement*
25:      $\mathcal{D} \leftarrow \left[(\chi, \sigma^i) \mapsto S\right]_{1 \leq \chi \leq n, \mathfrak{S}_\chi = \langle \mathbb{G}, \mathcal{L} \rangle, \sigma^i \in \mathcal{L}, \sigma_{|\sigma^i|}^i = \langle S, \chi \rangle}$
26:      $D \leftarrow \{(\kappa(\varsigma_{|\sigma^i|,k}^i), \chi) \mid \varsigma_{|\sigma^i|,k}^i \in \sigma_{|\sigma^i|}^i, \sigma^i \in \mathcal{L}, \mathfrak{S}_\chi = \langle \mathbb{G}, \mathcal{L} \rangle, 1 \leq \chi \leq n\}$
27:      refineEnd $\leftarrow$ REFINEATTEMPT($D, \mathcal{D}, \mathsf{End}, n$)
28:    **if not** refineEnd **then** $\Phi \leftarrow \Phi \cup \{\mathsf{End}(\mathsf{b}, \mathbf{true}) | \mathsf{End}(\mathsf{b}, \mathbf{true}) \in \bigcup_{1 \leq \chi \leq n} \Phi_\chi\}$

29:    refineEx $\leftarrow \exists \mathsf{a} \in \Sigma . \exists i, j \in \mathbb{N} . \mathsf{Exists}(\mathsf{a}, \mathbf{true}) \in \Phi_i \cup \Phi_j$
30:    **if** refineEx **then** $\triangleright$ Exists *refinement*
31:      $\mathcal{D} \leftarrow \left[(\chi, \sigma^i) \mapsto \left\{ \varpi(\varsigma_{j,\_}^i) \mid \varsigma_{j,\_}^i \in \sigma_j^i, \sigma^i \in \mathcal{L} \right\}\right]_{1 \leq \chi \leq n, \mathfrak{S}_\chi = \langle \mathbb{G}, \mathcal{L} \rangle}$
32:      $D \leftarrow \left\{ (\kappa(\varsigma_{j,k}^i), \chi) \mid \varsigma_{j,k}^i \in \sigma_j^i, \sigma^i \in \mathcal{L}, \mathfrak{S}_\chi = \langle \mathbb{G}, \mathcal{L} \rangle, 1 \leq \chi \leq n \right\}$
33:      refineEx $\leftarrow$ REFINEATTEMPT($D, \mathcal{D}, \mathsf{Exists}, n$)
34:    **if not** refineEx **then** $\Phi \leftarrow \Phi \cup \{\mathsf{Exists}(\mathsf{b}, \mathbf{true}) | \mathsf{Exists}(\mathsf{b}, \mathbf{true}) \in \bigcup_{1 \leq \chi \leq n} \Phi_\chi\}$
35:    **return** Φ

Algorithm 4 Ad Hoc Explainability: binary clause dataful refinement

---

1: **procedure** FILLINDATAFRAME(S_i^χ, φ)
　　▷ S_i^χ: *set of* **Vac***(uity),* **Viol***(ations), or* **Sat***(isfiability) conditions*
2:　　**global** DF
3:　　**if** $S_i^\chi = \emptyset$ **or** $S_i^\chi = $ Vac **then**
4:　　　　DF$[(\chi, i), \varphi] \leftarrow 0$
5:　　**else if** Viol $\in S_i^\chi$ **then**
6:　　　　DF$[(\chi, i), \varphi] \leftarrow -1$
7:　　**else**
8:　　　　DF$[(\chi, i), \varphi] \leftarrow 1$

9: **function** BINARYREFINE(A, B, $\mathfrak{G}_1, \ldots, \mathfrak{G}_n$; poly = **true**)
10:　　template $\leftarrow$ [ChainResponse, ChainPrecedence, Precedence, Response]
11:　　shorthands $\leftarrow$ [cr, cp, p, r]
12:　　dictionary $\leftarrow$ ZIP(template, shorthands)
13:　　pairs $\leftarrow \{\langle A, B\rangle, \langle B, A\rangle\}$
14:　　**for all** $1 \leq \chi \leq n$ **do**
15:　　　　$\mathfrak{G}_\chi = \langle \mathbb{G}_\chi, \mathscr{L}_\chi \rangle$
16:　　　　poly $\leftarrow$ **if** (poly **and** $\exists G_\psi \in \mathbb{G}_\chi.A, B \in N_{G_\psi}$) **then true else false**
17:　　　　CHAINS(A,B,θ,poly;$\mathscr{L}_\chi$); CHAINS(B,A,θ,poly;$\mathscr{L}_\chi$)　　　　▷ Algorithm 5
18:　　　　RESPPREC(A,B,θ,poly;$\mathscr{L}_\chi$); RESPPREC(B,A,θ,poly;$\mathscr{L}_\chi$)
19:　　**for all** $c_l(A', \textbf{true}, B', \textbf{true})$ s.t. $\langle A', B'\rangle \in$ pairs **and** $c_l \in$ template **do**
20:　　　　short $\leftarrow$ dictionary$[c_l]$
21:　　　　$D_{\text{act}} \leftarrow \left\{ (\kappa(\varsigma_{j,k}^i), \chi) \,\middle|\, \varsigma_{j,k}^i, \langle \varsigma_{j,k}^i, \_\rangle \in \text{act}(\chi)_{A',B'}^{\text{short}}, 1 \leq \chi \leq n \right\}$
22:　　　　**if** $D_{\text{act}} = \emptyset$ **then yield** $c_l(A', \textbf{true}, B', \textbf{true})$
23:　　　　**else**
24:　　　　　　$\mathcal{M} \leftarrow$ DECISIONTREE(D_{act})
25:　　　　　　**if** PURITY($\mathcal{M}$)$> 50\%$ **then**
26:　　　　　　　　**for all** $\pi \in \mathcal{M}$ **do**　　　　　　　　　　　▷ *Refine by activations*
27:　　　　　　　　　　**for all** $1 \leq \chi \leq n$, $\mathfrak{G}_\chi = \langle \_, \mathscr{L}_\chi \rangle$, $\varsigma_{j,k}^i, \langle \varsigma_{j,k}^i, \_\rangle \in \text{act}(\chi)_{A',B'}^{\text{short}}$ **do**
28:　　　　　　　　　　　　test $\leftarrow \pi(\kappa(\varsigma_{j,k}^i))$
29:　　　　　　　　　　　　**if** $\varsigma_{j,k}^i \in \text{viol}(\chi)_{A',B'}^{\text{short}}$ **then**
30:　　　　　　　　　　　　　　$S_i^\chi \leftarrow S_i^\chi \cup \{\text{test } \textbf{?} \text{ Viol} \textbf{:} \text{Vac}\}$
31:　　　　　　　　　　　　**else**
32:　　　　　　　　　　　　　　$S_i^\chi \leftarrow S_i^\chi \cup \{\text{test } \textbf{?} \text{ Sat} \textbf{:} \text{Vac}\}$
33:　　　　　　　　　　**for all** $\sigma^i \in \mathfrak{G}_\chi$ **do** FILLINDATAFRAME($S_i^\chi, c_l(A', \pi, B', \textbf{true})$)
34:　　　　　　**else**
35:　　　　　　　　$D_{\text{tgt}} \leftarrow \left\{ (\kappa(\varsigma_{h,k'}^i), \chi) \,\middle|\, \langle \varsigma_{j,k}^i, \varsigma_{h,k'}^i\rangle \in \text{act}(\chi)_{A',B'}^{\text{short}}, 1 \leq \chi \leq n \right\}$
36:　　　　　　　　$\mathcal{M} \leftarrow$ DECISIONTREE(D_{tgt})
37:　　　　　　　　**if** PURITY($\mathcal{M}$)$> 50\%$ **then**
38:　　　　　　　　　　**for all** $\pi \in \mathcal{M}$ **do**　　　　　　　　　　　▷ *Refine by targets*
39:　　　　　　　　　　　　**for all** $1 \leq \chi \leq n$, $\mathfrak{G}_\chi = \langle \_, \mathscr{L}_\chi \rangle$, $\varsigma_{j,k}^i, \langle \varsigma_{j,k}^i, \_\rangle \in \text{act}(\chi)_{A',B'}^{\text{short}}$ **do**
40:　　　　　　　　　　　　　　test $\leftarrow \pi(\kappa(\varsigma_{j,k}^i))$
41:　　　　　　　　　　　　　　**if** $\varsigma_{j,k}^i \in \text{viol}(\chi)_{A',B'}^{\text{short}}$ **then**
42:　　　　　　　　　　　　　　　　$S_i^\chi \leftarrow S_i^\chi \cup \{\text{test } \textbf{?} \text{ Viol} \textbf{:} \text{Vac}\}$
43:　　　　　　　　　　　　　　**else**
44:　　　　　　　　　　　　　　　　$S_i^\chi \leftarrow S_i^\chi \cup \{\text{test } \textbf{?} \text{ Sat} \textbf{:} \text{Vac}\}$
45:　　　　　　　　　　　　**for all** $\sigma^i \in \mathfrak{G}_\chi$ **do** FILLINDATAFRAME($S_i^\chi, c_l(A', \textbf{true}, B', \pi)$)
46:　　　　　　　　**else yield** $c_l(A', \textbf{true}, B', \textbf{true})$　　　▷ *Backtracking to the dataless clause*

---

tence can be easily formulated as propositional formulas composing Exists, we focus on refining the clauses that cannot be characterized through unary refinement alone: Precedence, Response, ChainPrecedence, and ChainResponse. We also discard ChainSuccession (and Succession) as those are composite clauses derivable from the former by conjunction, thus being also derivable from a propositional formula. For the time being, we do not consider $\text{All}\|c_l$ template variants, thus considering clause satisfaction if there are no violations across all activations within a trace.

After recalling that this procedure is invoked for any pair of activity labels A and B appearing as frequent in at least two segmented logs (Algorithm 2, Line 17), we first collect the evidence for the fulfilment of the aforementioned dataless clauses (narrated in the next paragraph). As this enables the collection of all the activated (Line 21) and targeted (Line 35) constituents' payloads, we can then see if is possible to extract a propositional characterization of the data through a decision tree; we consider the data summarization process through propositionalization successful if the tree achieves a suitable amount of purity (Lines 25 and 37). After extracting each path in such a tree as a binary predicate π, we consider all the activations first within a single trace σ^i of a segmented log $\mathfrak{S}_\chi$: differently from the refinement phases performed in DML [14] and in event-based mining algorithms [27], we change the clause satisfaction according to the joint fulfilment of the predicate π to generate a correct representation of the embedding: activated conditions that are no more satisfying π will always lead to a non-activation of the clause, and otherwise we retain the violation/satisfaction condition (Lines 28-32). We then consider the trace as violating the refined clause if at least one constituent leads to its violation (Line 28); we consider the clause vacuously satisfied if was never activated before or if the addition of the new π activation condition leads to de-activation of the clause testing (Line 4), and we consider as the trace being globally satisfied by the clause otherwise (Line 8). We draw similar considerations for the refinement of the target conditions. If neither of these two attempts at refining is satisfactory, we backtrack to the dataless clause (Line 45).

Algorithm 5 extends the mechanism in Bolt2 [11] in EMeriTAte for efficiently mining the aforementioned clauses by changing old trace IDs collections for clause activation (sat), violation (viol), or non-activation (vac), to considering the constituents generating these conditions. This reconstructs both the payloads associated with the activated and targeted conditions, and the constituent led to the violation of the clause. We denote all the possible target conditions for each activation leading to clause satisfaction as pairs of constituents associated with the activation. Bolt2 (e.g., Line 4 and Line 30) did not consider different overlapping constituents referring to the same MTS variable to avoid redundant and obvious correlations. This codebase can replicate this behaviour by setting the `poly` parameter to **false**. To recognize those, we now use activity label taxonomies and set the `poly` parameter to **true**: for constituents expressing distinct DT associated with the same MTS variable, we are interested in establishing temporal correlations between activated and targeted constituents if and only if the targeted (or activated) constituent terminates before the occurrence of the forthcoming activated (or targeted) one while always referring to the same variable. Bolt2 only considered pointwise non-polyadic events with no associated durative information, thus proving to be inadequate to support these new features.

Algorithm 5 Polyadic Mining for (Chain)Response and (Chain)Precedence.

1: **procedure** CHAINS$(\mathsf{A}, \mathsf{B}, \theta, \texttt{poly}; \mathcal{L}_y)$
2:    **for all** $\sigma^i \in \mathcal{L}_y$ **do**
3:       **for all** $\varsigma^i_{j,k}$ **s.t.** $\lambda(\varsigma^i_{j,k}) = \mathsf{A}$ **do**
4:          span $\leftarrow$ **if** $\texttt{poly}$ **then** $\pi(\varsigma^i_{j,k})$ **else** 1
5:          **if not** $\exists k.\lambda(\varsigma^i_{j+\mathsf{span},k}) = \mathsf{B}$ **then**$\{$ $\text{act}(y)^{\mathsf{cr}}_{\mathsf{A},\mathsf{B}}.\text{add}(\varsigma^i_{j,k})$; $\text{viol}(y)^{\mathsf{cr}}_{\mathsf{A},\mathsf{B}}.\text{add}(\varsigma^i_{j,k})\}$
6:          **else** $\text{act}(y)^{\mathsf{cr}}_{\mathsf{A},\mathsf{B}}.\text{add}(\langle \varsigma^i_{j,k}, \varsigma^i_{j+\mathsf{span},k} \rangle)$ **for all** k **s.t.** $\lambda(\varsigma^i_{j+\mathsf{span},k}) = \mathsf{B}$.
7:          **if** $j > 1$ **then**
8:             $\text{act}(y)^{\mathsf{cp}}_{\mathsf{A},\mathsf{B}}.\text{add}(\varsigma^i_{j,k})$;
9:             **if not** $\exists h, k'.\lambda(\varsigma^i_{h,k'}) = \mathsf{B}$ **and** $h + (\texttt{poly}\,?\,\delta(\varsigma^i_{h,k'}):1) \leq j$ **then**
10:                $\text{viol}(y)^{\mathsf{cp}}_{\mathsf{A},\mathsf{B}}.\text{add}(\varsigma^i_{j,k})$
11:             **else** $\text{act}(y)^{\mathsf{cr}}_{\mathsf{A},\mathsf{B}}.\text{add}(\langle \varsigma^i_{j,k}, \varsigma^i_{h,k'} \rangle)$ **for all** h, k' **s.t.** $\lambda(\varsigma^i_{h,k'}) = \mathsf{B}$ **and** $h + (\texttt{poly}\,?\,\delta(\varsigma^i_{h,k'}):1) \leq j$

12: **procedure** RESPPREC$(\mathsf{A}, \mathsf{B}, \theta, \texttt{poly}; \mathcal{L}_y)$
13:    **for all** $\sigma^i \in \mathcal{L}_y$ **do**
14:       **if not** $\exists \varsigma^i_{h,k}.\lambda(\varsigma^i_{h,k}) = \mathsf{A}$ **then**
15:          **return**   ▷ *Not reporting vacuous satisfaction explicitly*
16:       **else if not** $\exists \varsigma^i_{h,k}.\lambda(\varsigma^i_{h,k}) = \mathsf{B}$ **then**
17:          **for all** $\varsigma^i_{j,k}$ **s.t.** $\lambda(\varsigma^i_{j,k}) = \mathsf{A}$ **do**
18:             $\text{act}(y)^{\mathsf{r}}_{\mathsf{A},\mathsf{B}}.\text{add}(\varsigma^i_{j,k})$; $\text{viol}(y)^{\mathsf{r}}_{\mathsf{A},\mathsf{B}}.\text{add}(\varsigma^i_{j,k})$; $\text{act}(y)^{\mathsf{p}}_{\mathsf{A},\mathsf{B}}.\text{add}(\varsigma^i_{j,k})$;
19:       **else**
20:          **for all** $\sigma^i \in \mathcal{L}_y$ **s.t.** $\exists \varsigma^i_{j,k}.\,\lambda(\varsigma^i_{j,k}) = \mathsf{A} \vee \lambda(\varsigma^i_{j,k}) = \mathsf{B}$ **do**
21:             **if not** $\exists h, k'.\,\lambda(\varsigma^i_{h,k'}) = \mathsf{B}$ **then**   ▷ *Only* A*s*
22:                $\text{act}(y)^{\mathsf{r}}_{\mathsf{A},\mathsf{B}}.\text{add}(\varsigma^i_{j,k})$; $\text{viol}(y)^{\mathsf{r}}_{\mathsf{A},\mathsf{B}}.\text{add}(\varsigma^i_{j,k})$;
23:                $\text{act}(y)^{\mathsf{p}}_{\mathsf{A},\mathsf{B}}.\text{add}(\varsigma^i_{j,k})$; $\text{viol}(y)^{\mathsf{p}}_{\mathsf{B},\mathsf{A}}.\text{add}(\varsigma^i_{j,k})$;
24:             **else if not** $\exists h, k'.\,\lambda(\varsigma^i_{h,k'}) = \mathsf{A}$ **then**   ▷ *Only* B*s*
25:                $\text{act}(y)^{\mathsf{p}}_{\mathsf{B},\mathsf{A}}.\text{add}(\varsigma^i_{j,k})$; $\text{viol}(y)^{\mathsf{p}}_{\mathsf{A},\mathsf{B}}.\text{add}(\langle \texttt{NULL}, \varsigma^i_{j,k} \rangle)$
26:             **else if** $\lambda(\varsigma^i_{j,k}) = \mathsf{A}$ **then**   ▷ *Both occur*
27:                $\text{act}(y)^{\mathsf{p}}_{\mathsf{A},\mathsf{B}}.\text{add}(\sigma^i_{j,k})$;
28:                **if** $\exists \varsigma^i_{j+h,k}, \varsigma^i_{j,k'}.\,h \geq 0$ **and** $\lambda(\varsigma^i_{j+h,k}) = \mathsf{B}$ **and** $\lambda(\varsigma^i_{j,k'}) = \mathsf{A}$ **then**
29:                   $\text{viol}(y)^{\mathsf{p}}_{\mathsf{B},\mathsf{A}}.\text{add}(\varsigma^i_{j,k})$
30:                **if not** $\exists h, k'.\,\lambda(\varsigma^i_{h,k'}) = \mathsf{B}$ **and** $j + (\texttt{poly}\,?\,\pi(\varsigma^i_{j,k}):1) \leq h$ **then**
31:                   $\text{viol}(y)^{\mathsf{r}}_{\mathsf{A},\mathsf{B}}.\text{add}(\varsigma^i_{j,k})$
32:                **else** $\text{act}(y)^{\mathsf{r}}_{\mathsf{A},\mathsf{B}}.\text{add}(\langle \varsigma^i_{j,k}, \varsigma^i_{h,k'} \rangle)$ **for all** h, k' **s.t.** $\lambda(\varsigma^i_{h,k'}) = \mathsf{B}$ **and** $j + (\texttt{poly}\,?\,\pi(\varsigma^i_{j,k}):1) \leq h$

---

5. Empirical Results

We exploit some time series datasets made available through the `sktime` library [29] while considering the Dyskinetic event dataset from our previous contribution [16]. Italy Power Demand [26] provides a dataset of univariate and equal-length time series, where the classification label is used to differentiate energy consumption patterns in the October-March period vs the April-September ones. The Basic Motions dataset[9] considers multivariate time series with 6 dimensions and distinguishing different motion types, walking, resting, running, and badminton. This dataset collects the x, y, and z axis information from wrist-watch accelerometer and a gyroscope; this dataset has been used to compare this motion dataset with the one discussed in our previous work while remarking the inherent more difficult nature of solving this other clinical problem. The Dyskinetic dataset [16] attempts at categorize Dyskinetic/Off events in terms of different drug assumption patterns as well as motor sensors information. Differently from the previous set of experiments [16], we now consider for both our algorithms and the competing MTSC approaches the full dataset also considering the active principles assumption intakes estimated using a rough approximation from current literature [6]. We also use the OSULeaf [21] dataset to remark how a simple image classification problem can be also represented as a time series classification problem: the univariate series were obtained by color image segmentation and boundary extraction (in the anti-clockwise direction) from digitized leaf images of six classes: Acer Circinatum, Acer Glabrum, Acer Macrophyllum, Acer Negundo, Quercus Garryana, and Quercus Kelloggii.

We consider such datasets for the following reasons: despite EMeriTAte was explicitly designed to capture trend correlations across different dimensions, we also use univariate datasets such as Italy Power Demand and OsuLeaf to show that this approach can be also applied to simpler dataset. Furthermore, we consider another multivariate dataset also considering different motor sensors to showcase the peculiar difference between the problem posed by the identification of the Dyskinetic Events from simply detecting different movement patterns. We provide some general statistics for these in Table 3.

Table 3. Dataset statistics

Dataset	$|D|$	avg_T Segments	#Dimensions	$avg_{\mathfrak{E}}$	$|\mathfrak{E}|$	$|\Sigma|$	#Classes
Italy Power Demand	1096	1	1	24	13		2
Basic Motions	80	1	6	100	72		4
OsuLeaf	442	1	1	427	12		6
Dyskinetic Events	2	23	30	44	436		2

Our benchmarks were conducted on a Dell Precision mobile workstation 5760 running Ubuntu 22.04. The specifications of the machine include an Intel® Xeon(R) W-11955M CPU @ 2.60GHz × 16, 64GB DDR4 3200MHz RAM, with 500 GB of free disk space.

---

[9] http://timeseriesclassification.com/description.php?Dataset=BasicMotions

5.1.   Run-time efficiency of EMeriTAte vs EMeriTAte+DF

Fig. 2 refers to the running times of both our proposed MTSC algorithms, from both our previous contribution and from the current paper. The first three phases refer to the A Priori explainability segment, while the last two refer to the Ad Hoc one. In both scenarios, we consider $\theta = 0$ for maximising the DECLARE recall, and consider in both circumstances a sensitivity parameter of $\varepsilon = 10^{-4}$. We considered the time required to fully process each dataset.
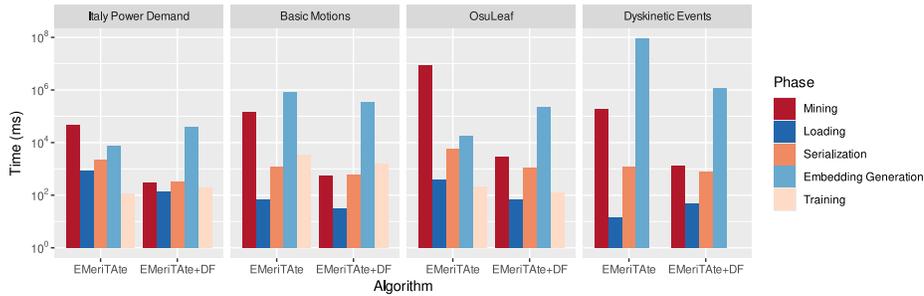
Fig. 2. Fine-Grained Mining+Learning times for the training phase of both EMeriTAte and EMeriTAte+DF.

Although the loading times of the datasets are comparable, we observe that the loading phase in this novel solution, which is different from the former, also includes a time series indexing phase. This then leads to a significant improvement in time minimisation, also ascribable to discarding some redundant DT activity labels. This is also witnessed in the serialization phase, which has slightly decreased despite the addition of catch22 payloads. Running times for the mining phase in EMeriTAte+DF remark that this algorithm is heavily dominated by the number of traces and segments being available as, in these situations, the former version of the algorithm was heavily hampered by conducting the mining and the embedding phases in two different times, thus requiring to load the data and index it several times. The newly proposed approach might have a more significant overhead in some situations. In fact, generating a dataful representation of the clauses requires accessing the constituents' payloads several times, thus increasing the number of times we access KnoBAB [11] for retrieving log information. Notwithstanding the former, this leads to a global running time decrease (Fig. 3).

5.2.   Comparing competing MTSC approaches

We used the Python `sktime` library to ensure a uniform implementation for all competing approaches discussed in Section 2.3, for which we used the default hyperparameters setup from `sktime`. As all of the competing classifiers do not support classification of time series of different lengths, and given that these classifiers do not support time series varying their classification outcome in time, for the sole Dyskinetic event dataset, we consider the maximal projections belonging to the same class and, between each of these, we
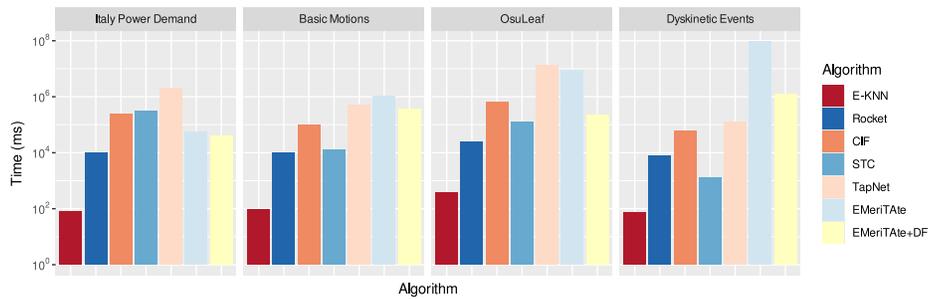
Fig. 3. Comparing the cumulative training times

consider multiple sliding windows of the same size. This overall builds up the number of time series to be classified by competing approaches.

Run-time efficiency. Fig. 3 compares the running time for all the MTSC approaches being trained over all the aforementioned datasets without splitting the data into training and testing datasets, thus using the same setting for the previous experiments. Competing approaches are not significantly impacted over the number of MTS dimensions as our approaches: please observe the tunning times for the Dyskinetic events where, despite the number of traces increased so to obtain multiple MTS of the same length, the competing approaches still provide a lesser running time if compared to ours. This is motivated by our solutions testing multiple possible correlations across dimension and activity labels, while other solutions such as Rocket and TapNet, attempt to capture cross-dimensional correlations through convolutions, thus flattening up the values via dimensionality reduction. Similar considerations can also be carried out by the other classifiers, which have not been specifically designed to consider correlations across dimensions: Euclidean distance for MTS only considers dimensionality-wise similarity, CIF considers variations within one fixed time and length sliding window while considering variations only within one single dimension, and STC considers shapelets only occurring within one single dimension. On the other hand, TapNet is heavily dominated by the number of traces being considered at training time and, across all the competitors, is the one always requiring a considerable amount on training time. EMeriTAte solution overtakes EMeriTAte+DF over the Dyskinetic dataset, while the other dataset consistently shows similar running times.

Classification outcomes. Fig. 4 provides the accuracy, precision, recall, and F1 score results across all the aforementioned datasets and competing approaches when testing the solutions with a 70%-30% split between training and testing dataset using stratified k-fold sampling. For both EMeriTAte and EMeriTAte+DF, we consider training the Decision Trees over the trace embeddings with a maximum depth of 5, so to minimise the chance of generating models overfitting the data. We run 10 experiments, for which we report the average score and the distance between the greatest and the lowest value obtained ($\pm$). When considering datasets with more than two classes, we use macro measures.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
E-KNN	96.47 ± 1.22	97.36 ± 1.19	95.58 ± 1.82	96.45 ± 1.24
Rocket [18]	97.02 ± 1.06	96.70 ± 1.72	97.39 ± 1.21	97.04 ± 1.05
CIF [32]	96.69 ± 1.82	96.73 ± 1.75	96.67 ± 3.03	96.69 ± 1.87
STC [23]	96.81 ± 1.37	96.81 ± 1.70	96.85 ± 3.03	96.81 ± 1.42
TapNet [43]	96.20 ± ε	94.76 ± 1.62	97.88 ± 0.91	96.28 ± ε
EMeriTAte [16]	99.59 ± ε	99.73 ± ε	99.45 ± 0.01	99.59 ± ε
EMeriTAte+DF	96.23 ± 1.52	96.24 ± 1.51	96.23 ± 1.52	96.23 ± 1.52

(a) Italy Power Demand

Algorithm	Accuracy (%)	M. Precision (%)	M. Recall (%)	M. F1 (%)
E-KNN	62.50 ± 8.33	53.09 ± 3.79	62.50 ± 8.33	53.91 ± 7.15
Rocket [18]	99.58 ± 2.08	99.64 ± 1.79	99.58 ± 2.08	99.58 ± 2.10
CIF [32]	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00
STC [23]	99.58 ± 2.08	99.64 ± 1.79	99.58 ± 2.08	99.58 ± 2.10
TapNet [43]	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00
EMeriTAte [16]	91.66 ± 8.33	92.69 ± 7.81	91.66 ± 8.33	91.51 ± 8.57
EMeriTAte+DF	98.75 ± 2.08	98.92 ± 1.78	98.75 ± 2.08	98.74 ± 2.10

(b) Basic Motions

Algorithm	Accuracy (%)	M. Precision (%)	M. Recall (%)	M. F1 (%)
E-KNN	63.46 ± 6.02	65.43 ± 5.79	62.59 ± 6.53	62.98 ± 6.54
Rocket [18]	96.32 ± 2.26	96.77 ± 1.98	95.98 ± 2.41	96.25 ± 2.24
CIF [32]	84.44 ± 3.01	86.55 ± 3.25	82.38 ± 4.94	83.29 ± 4.55
STC [23]	92.86 ± 3.01	93.35 ± 3.23	91.72 ± 3.78	92.25 ± 3.56
TapNet [43]	82.71 ± 1.50	88.29 ± 1.06	78.56 ± 5.49	78.59 ± 6.65
EMeriTAte[16]	79.62 ± 8.27	83.72 ± 7.78	80.13 ± 8.50	80.39 ± 8.56
EMeriTAte+DF	99.70 ± 0.01	99.63 ± 0.01	99.70 ± ε	99.65 ± ε

(c) OsuLeaf

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
E-KNN	12.14 ± 10.71	10.06 ± 18.18	12.86 ± 28.57	11.21 ± 22.22
Rocket [18]	12.14 ± 10.71	7.80 ± 18.18	10.00 ± 28.57	8.65 ± 22.22
CIF [32]	12.14 ± 10.71	4.58 ± 16.67	5.71 ± 21.43	5.08 ± 18.75
STC [23]	25.71 ± 17.86	14.44 ± 22.22	12.86 ± 28.57	13.02 ± 25.00
TapNet [43]	15.00 ± 10.71	15.19 ± 18.18	20.0 ± 28.57	17.09 ± 22.22
EMeriTAte [16]	98.57 ± 7.14	100.0 ± 0.00	97.14 ± 14.28	98.33 ± 8.33
EMeriTAte+DF	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00

(d) Dyskinetic Events

Fig. 4. Training Results over the datasets of interest. Macro metrics are used for datasets containing more than 2 classes. Numbers in blue (red) remark the best (worst) results

Concerning the Dyskinetic dataset, we kept different training/testing splits from the ones in our previous paper.

We observe that, for three datasets out of four, EMeriTAte+DF outperforms our previous implementation. The major score distance between these two solutions can be found on the OsuLeaf dataset: this can be explained as the main difference between the two solutions resides in the usage of the data refinement, while the discard of DT patterns seems not to have interfered with the classification outcome. For the Italy Power Demand dataset, we see that our first solution outperforms the current implementation: given the above, this can be ascribed to the presence of several fluctuations events which were not discarded in our previous solution.

By comparing our solutions to the competing approaches, no competitor can constantly outperform the others across all the datasets and, when EMeriTAte+DF did not achieve maximum scores, it still achieved $\approx 99\%$ of accuracy, macro precision, macro recall, and F1 score. E-KNN scores low values for datasets such as the former and OsuLeaf, thus remarking the impracticality of merely considering time series similarity as a valid pathway for classification outcomes for real-world datasets.

While comparing the results over the Dyskinetic dataset, it is clear that the best approaches were the ones considering silhouette (STC) and attention ones (TapNet) while still scoring metrics way below 50%. This can be motivated as follows: either the process of splitting the time series lost some important correlation information or truncated some patterns, or these solutions still cannot account for extracting the best behavioural predictors for the dataset of interest. After analysing the models extracted from EMeriTAte+DF, we can clearly see that the classification always refers to correlations across different dimensions. Given the explanation being extracted, this then remarks that the dataset of interest is heavily characterized by correlations across dimensions that could not be adequately captured by the pre-existing solutions.

6.   Conclusion and Future Works

This paper provides the first algorithm towards a generalization of existing algorithms in the literature for classifications of temporal behaviours: in fact, we use a purely event-driven technique to classify multivariate numerical temporal sequences. By doing so, we have shown how it is possible to appropriately discretize a numerical representation, maintaining as much of the information in the data as possible. To appropriately characterize the numerical data, we have also deemed it necessary to further generalize the representation of temporal data to events, prescribing that multiple distinct events of different durations can occur at a given instant of time. From the generalization of the deviance mining algorithms, we obtain a new classifier that can be used to classify numerical time series with good results in precision, recall, and accuracy. Subsequent studies will also establish this mechanism's validity in characterizing event-based data.

Future works will investigate whether considering additional events not currently mined in the indexed DT phase will make EMeriTAte+DF reach the same precision, recall, and accuracy results from our previous contribution, EMeriTAte, without tampering the running time of the subsequent *ad hoc* phase. This will be supported by applying the experiments on novel datasets, while investigating whether the current solution is suitable to address the aforementioned challenge. Despite preliminary results remark that the

non-negligible running time of this latter phase might be due to the considerable presence of events to ascertain, considering larger amounts of data will necessarily require further improving over the presently proposed algorithms. As both our solutions are implemented using KnoBAB [11] as a temporal-driven database, future works will also require to extend it so to move from a in-memory data representation to a one using secondary memory. We will also consider the possibility of applying the general-to-specific mining approach that we previously investigated over Bolt2 for mining DT patterns, so to achieve mining efficiency without resorting to discarding noisy or redundant events.

Experiments over the datasets from Table 3 lead dataful specifications describing the classes showing only refinements over the activation conditions, but not on the target ones. We will also test the possibility of other datasets also requiring the definition of correlation conditions between activated and targeted events, thus requiring the definition of comparisons between activated and targeted constituents' payloads. Despite the possibility of doing so though weighted oblique trees [42], this decision will come at the cost of significantly increasing computational overhead over the refinement phase. As a byproduct of this, further work needs also to be done to further improve over the fitting of oblique decision trees.

Last, will also explore the possibility of time series forecasting from the mined and learned model. Given the dualism between polyadic traces and MTS and given the possibility of generating dataful traces out of conjunctive models [9], it should then be required to unpack the catch22 values being generated to reconstruct actual time series values. Thus, future works will need to address this challenge and verify its possibility. After achieving this, a further extension will be required to generate traces out of declarative models expressed as arbitrary propositions. We will then need to compare such a solution with state-of-the-art approaches in both machine learning [34,41,33] and statistical [40,28] literature.

References

1. van der Aalst, W.M.P.: Discovering Directly-Follows Complete Petri Nets from Event Data, pp. 539–558. Springer Nature Switzerland, Cham (2022), `doi.org/10.1007/978-3-031-15629-8_29`

2. Acampora, G., Vitiello, A., Di Stefano, B., van der Aalst, W., Gunther, C., Verbeek, E.: IEEE 1849: The XES Standard: The Second IEEE Standard Sponsored by IEEE Computational Intelligence Society. IEEE Comp. Int. Mag. 12(2), 4–8 (2017), `doi.org/10.1109/MCI.2017.2670420`

3. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM 26(11), 832–843 (Nov 1983), `doi.org/10.1016/B978-1-4832-1447-4.50033-X`

4. Andrzejewski, W., Bębel, B., Boiński, P., Wrembel, R.: On tuning parameters guiding similarity computations in a data deduplication pipeline for customers records: Experience from a r&d project. Information Systems 121, 102323 (2024), `https://www.sciencedirect.com/science/article/pii/S030643792300159X`

5. Anselma, L., Bottrighi, A., Montani, S., Terenziani, P.: Extending BCDM to cope with proposals and evaluations of updates. IEEE Transactions on Knowledge and Data Engineering 25(3), 556–570 (2013), `doi.org/10.1109/TKDE.2011.170`

6. Arav, Y., Zohar, A.: Model-based optimization of controlled release formulation of levodopa for Parkinson's disease. Scientific Reports 13(1), 15869 (Sep 2023), `doi.org/10.1038/s41598-023-42878-5`

7. Avolio, M., Fuduli, A., Vocaturo, E., Zumpano, E.: On detection of diabetic retinopathy via multiple instance learning. In: Proceedings of the International Database Engineered Applications Symposium Conference, IDEAS 2023, Heraklion, Crete, Greece, May 5-7, 2023. pp. 170–176. ACM (2023), `doi.org/10.1145/3589462.3589490`

8. Bergami, G.: On efficiently equi-joining graphs. In: Proceedings of the 25th International Database Engineering & Applications Symposium. p. 222–231. IDEAS '21, Association for Computing Machinery, New York, NY, USA (2021), `doi.org/10.1145/3472163.3472269`

9. Bergami, G.: Fast synthetic data-aware log generation for temporal declarative models. In: Hartig, O., Yoshida, Y. (eds.) Proceedings of the 6th Joint Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), Seattle, WA, USA, 18 June 2023. pp. 7:1–7:9. ACM (2023), `https://doi.org/10.1145/3594778.3594881`

10. Bergami, G.: DECLAREd: A Polytime LTL_f Fragment. Logics 2(2), 79–111 (2024), `doi.org/10.3390/logics2020004`

11. Bergami, G., Appleby, S., Morgan, G.: Specification mining over temporal data. Computers 12(9) (2023), `doi.org/10.3390/computers12090185`

12. Bergami, G., Bertini, F., Montesi, D.: Hierarchical embedding for DAG reachability queries. In: Proceedings of the 24th Symposium on International Database Engineering & Applications. IDEAS '20, Association for Computing Machinery, New York, NY, USA (2020), `doi.org/10.1145/3410566.3410583`

13. Bergami, G., Fox, O.R.: How explainable really is AI? benchmarking explainable AI. Logics 3(3) (2025), `doi.org/10.3390/logics3030009`

14. Bergami, G., Francescomarino, C.D., Ghidini, C., Maggi, F.M., Puura, J.: Exploring business process deviance with sequential and declarative patterns. CoRR abs/2111.12454 (2021), `doi.org/10.48550/arXiv.2111.12454`

15. Bergami, G., Maggi, F.M., Montali, M., Peñaloza, R.: Probabilistic trace alignment. In: 2021 3rd International Conference on Process Mining (ICPM). pp. 9–16 (2021), `doi.org/10.1109/ICPM53251.2021.9576856`

16. Bergami, G., Packer, E., Scott, K., Din, S.D.: Predicting dyskinetic events through verified multivariate time series classification. In: Chbeir, R., Ilarri, S., Manolopoulos, Y., Revesz, P.Z., Bernardino, J., Leung, C.K. (eds.) Database Engineered Applications - 28th International Symposium, IDEAS 2024, Bayonne, France, August 26-29, 2024, Proceedings. Lecture Notes in Computer Science, vol. 15511, pp. 49–62. Springer (2024), `doi.org/10.1007/978-3-031-83472-1_4`

17. De Giacomo, G., Maggi, F.M., Marrella, A., Patrizi, F.: On the Disruptive Effectiveness of Automated Planning for LTL_f-Based Trace Alignment. In: AAAI'17. AAAI press (2017), `doi.org/10.1609/aaai.v31i1.11020`

18. Dempster, A., Petitjean, F., Webb, G.I.: ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min. Knowl. Discov. 34(5), 1454–1495 (2020), `doi.org/10.1007/s10618-020-00701-z`

19. Donzé, A.: On signal temporal logic. In: Runtime Verification - 4th International Conference, RV 2013, Rennes, France, September 24-27, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8174, pp. 382–383. Springer (2013), `doi.org/10.1007/978-3-642-40787-1_27`

20. Färber, M.: Denotational semantics and a fast interpreter for jq. CoRR abs/2302.10576 (2023), `https://doi.org/10.48550/arXiv.2302.10576`

21. Gandhi, A.: Content-Based Image Retrieval: Plant Species Identification. Master's thesis, Oregon State University (2002), `https://web.engr.oregonstate.edu/~tgd/leaves/acer-quercus.pdf`

22. Gellér, Z., Kurbalija, V., Ivanović, M.: FAP: A time series analysis and mining framework for scientific and practical applications. Comput. Sci. Inf. Syst. Forthcoming (2025)

23. Hills, J., Lines, J., Baranauskas, E., Mapp, J., Bagnall, A.: Classification of time series by shapelet transformation. Data Mining and Knowledge Discovery 28(4), 851–881 (Jul 2014), `doi.org/10.1007/s10618-013-0322-1`

24. Huo, X., Hao, K., Chen, L., song Tang, X., Wang, T., Cai, X.: A dynamic soft sensor of industrial fuzzy time series with propositional linear temporal logic. Expert Systems with Applications 201, 117176 (2022), `doi.org/10.1016/j.eswa.2022.117176`

25. Inoue, T., Kubota, K., Ikami, T., Egami, Y., Nagai, H., Kashikawa, T., Kimura, K., Matsuda, Y.: Clustering method for time-series images using quantum-inspired digital annealer technology. Communications Engineering 3(1), 10 (Jan 2024), `https://doi.org/10.1038/s44172-023-00158-0`

26. Keogh, E., Wei, L., Xi, X., Lonardi, S., Shieh, J., Sirowy, S.: Intelligent icons: Integrating lite-weight data mining and visualization into GUI operating systems. In: Sixth International Conference on Data Mining (ICDM'06). pp. 912–916 (2006), `doi.org/10.1109/ICDM.2006.90`

27. Leno, V., Dumas, M., Maggi, F.M., La Rosa, M., Polyvyanyy, A.: Automated discovery of declarative process models with correlated data conditions. Information Systems 89, 101482 (2020), `doi.org/10.1016/j.is.2019.101482`

28. Livera, A.M.D., Hyndman, R.J., Snyder, R.D.: Forecasting time series with complex seasonal patterns using exponential smoothing. Journal of the American Statistical Association 106(496), 1513–1527 (2011), `doi.org/10.1198/jasa.2011.tm09771`

29. Löning, M., Bagnall, A.J., Ganesh, S., Kazakov, V., Lines, J., Király, F.J.: sktime: A unified interface for machine learning with time series. CoRR abs/1909.07872 (2019), `doi.org/10.48550/arXiv.1909.07872`

30. Lubba, C.H., Sethi, S.S., Knaute, P., Schultz, S.R., Fulcher, B.D., Jones, N.S.: catch22: CAnonical Time-series CHaracteristics - selected through highly comparative time-series analysis. Data Min. Knowl. Discov. 33(6), 1821–1852 (2019), `doi.org/10.1007/S10618-019-00647-X`

31. Wes McKinney: Data Structures for Statistical Computing in Python. In: Stéfan van der Walt, Jarrod Millman (eds.) Proceedings of the 9th Python in Science Conference. pp. 56 – 61 (2010)

32. Middlehurst, M., Large, J., Bagnall, A.: The canonical interval forest (CIF) classifier for time series classification. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 188–195 (Dec 2020), `doi.org/10.1109/BigData50022.2020.9378424`

33. Olivares, K.G., Challú, C., Garza, A., Canseco, M.M., Dubrawski, A.: NeuralForecast: User friendly state-of-the-art neural forecasting models. PyCon Salt Lake City, Utah, US 2022 (2022), `https://github.com/Nixtla/neuralforecast`

34. Pecev, P., Rackovic, M.: LTR-MDTS structure - a structure for multiple dependent time series prediction. Comput. Sci. Inf. Syst. 14(2), 467–490 (2017), `doi.org/10.2298/CSIS150815004P`

35. Petermann, A., Micale, G., Bergami, G., Pulvirenti, A., Rahm, E.: Mining and ranking of generalized multi-dimensional frequent subgraphs. In: Twelfth International Conference on Digital

Information Management, ICDIM 2017, Fukuoka, Japan, September 12-14, 2017. pp. 236–245. IEEE (2017), `doi.org/10.1109/ICDIM.2017.8244685`

36. Pešić, M., Schonenberg, H., van der Aalst, W.M.P.: DECLARE: full support for loosely-structured processes. In: EDOC. pp. 287–300. IEEE Computer Society (2007), `doi.org/10.1109/EDOC.2007.14`

37. Rost, C., Gómez, K., Täschner, M., Fritzsche, P., Schons, L., Christ, L., Adameit, T., Junghanns, M., Rahm, E.: Distributed temporal graph analytics with GRADOOP. VLDB J. 31(2), 375–401 (2022), `doi.org/10.1007/s00778-021-00667-4`

38. Ruiz, A.P., Flynn, M., Large, J., Middlehurst, M., Bagnall, A.: The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery 35(2), 401–449 (Mar 2021), `doi.org/10.1007/s10618-020-00727-3`

39. Seshia, S.A., Sadigh, D., Sastry, S.S.: Toward verified artificial intelligence. Commun. ACM 65(7), 46–55 (Jun 2022), `doi.org/10.1145/3503914`

40. Swain, S., Nandi, S., Patel, P.: Development of an ARIMA model for monthly rainfall forecasting over Khordha District, Odisha, India. In: Sa, P.K., Bakshi, S., Hatzilygeroudis, I.K., Sahoo, M.N. (eds.) Recent Findings in Intelligent Computing Techniques. pp. 325–331. Springer Singapore, Singapore (2018), `doi.org/10.1007/978-981-10-8636-6_34`

41. Tang, H., Zhang, C., Jin, M., Yu, Q., Wang, Z., Jin, X., Zhang, Y., Du, M.: Time series forecasting with llms: Understanding and enhancing model capabilities. SIGKDD Explor. Newsl. 26(2), 109–118 (Jan 2025), `doi.org/10.1145/3715073.3715083`

42. Yang, B.B., Shen, S.Q., Gao, W.: Weighted oblique decision trees 33(01), 5621–5627, `doi.org/10.1609/aaai.v33i01.33015621`

43. Zhang, X., Gao, Y., Lin, J., Lu, C.: TapNet: Multivariate time series classification with attentional prototypical network 34(4), 6845–6852 (2020), `doi.org/10.1609/aaai.v34i04.6165`

Giacomo Bergami, at the time of the writing, is a Lecturer (Assistant Professor) in Game Technology at Newcastle University, and is the PI for the LOGic for Data Science (LogDS) group. We advocate that Logic should be used as a main driving force for Data Science to obtain Verified and Explainable Artificial Intelligence. He received the Bachelor's and Master's Degree in Computer Science with full marks (both with a final passing grade of 110/110 with Merit). From the 1st of November 2014 to the 1st of November 2017, my PhD dissertation at Bologna University, Italy, CSE Dept. was supervised by Prof. Danilo Montesi: the research focused on two novel graph operators: graph joins (for combining data) and graph nesting (for structural aggregations), contextualised within a clinical data integration setting. Before obtaining my lectureship position, I gained international experience by working abroad, both as a Visiting PhD Student (Universität Leipzig, Germany) and as a PostDoc Researcher (University of Florida, USA).

Kirsty Scott is a Postdoctoral Research Associate at Newcastle University since 2023, where she works in the Brain and Movement Team. She got the PhD in Mechanical Engineering at the University of Sheffield.

Emma Packer is a PhD student at Newcastle University. Emma's PhD explores how medication influences everyday walking in people with Parkinson's. Emma has an undergraduate degree in Biomedical Sciences, and a masters degree in Neuroscience.

Silvia Del Din is a bioengineer by background and a Reader (Associate Professor) in Digital Healthcare for translational applications at Newcastle University. She has attracted >£2.5 Million in funding as PI and Co-I of various digital health projects (e.g. from MRC, EPSRC, IMI-EU (Mobilise-D, IDEA-FAST)). She leads the wearable/digital health technology team within the Brain and Movement Research Group where she has contributed to build the digital health theme. Her research aims at innovating digital healthcare and translating it into clinical practice. Her expertise focuses on enhancing the use of digital health technology together with innovative data analysis techniques to enable remote monitoring and facilitate clinical management in ageing and neurodegenerative disorders (e.g. Parkinson's disease). She has published 100 full papers. She is part of the Editorial Board of the IEEE Open Journal of Engineering in Medicine and Biology and an Associate Editor for the IEEE Journal of Biomedical and Health Informatics. She serves as a member on national and international scientific committees (e.g., Parkinson's UK College of Experts, ISPGR Scientific Content Committee); since 2022 she is the elected Chair of the Early Career Researchers Working Group of the BioMedEng Association Council.