Shape Expressions for Specifying and Extracting Signal Features

Dejan Ničković[1], Xin Qin[2], Thomas Ferrère[3], Cristinel Mateis[1], and Jyotirmoy Deshmukh[2]

[1] AIT Austrian Institute of Technology
[2] University of Southern California
[3] IST Austria

Abstract. Cyber-physical systems (CPS) and the Internet-of-Things (IoT) result in a tremendous amount of generated, measured and recorded time-series data. Extracting temporal segments that encode patterns with useful information out of these huge amounts of data is an extremely difficult problem. We propose *shape expressions* as a declarative formalism for specifying, querying and extracting sophisticated temporal patterns from possibly noisy data. Shape expressions are regular expressions with arbitrary (linear, exponential, sinusoidal, etc.) shapes with parameters as atomic predicates and additional constraints on these parameters. We equip shape expressions with a novel *noisy* semantics that combines regular expression matching semantics with statistical regression. We characterize essential properties of the formalism and propose an efficient approximate shape expression matching procedure. We demonstrate the wide applicability of this technique on two case studies.

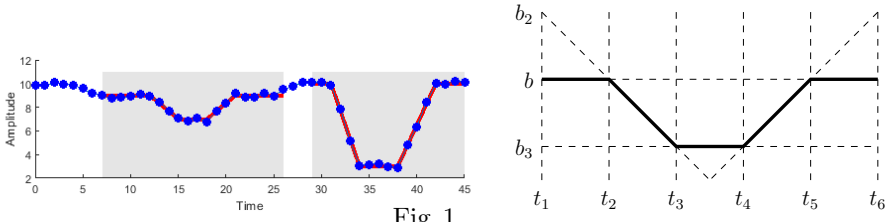
1 Introduction

Cyber-physical systems (CPS) and Internet-of-Things (IoT) applications are everywhere around us - smart buildings that adapt heating control to the user's habit, intelligent transportation systems that optimize traffic based on the continuous monitoring of the road conditions, wearable health monitoring devices, and medical devices that fine-tune a given therapy depending on sensing a patient's health. These applications are inherently data-driven – the decisions of the system rely on the measurement and analysis of the dynamic behavior of the environment. Low-cost sensing solutions combined with the availability of powerful edge and cloud devices to store and process data has led to a tremendous increase in the generation, measurement and recording of time-series data. Processing these huge streams of available data in an efficient manner to extract useful information is challenging. It is often the case that only specific segments of the time series contain interesting and relevant patterns. For instance, an electricity provider may be interested in observing spikes or oscillations in the voltage signals. A medical device manufacturer may want to detect anomalous cardiac behavior. A wearable device maker would like to associate specific patterns in the measurements from accelerometer and gyroscope sensors to a concrete user activity, such as running or walking.

Such patterns can be often characterized with geometric shapes observed in the time-series data; e.g., a spike can be specified as an "upward triangle", i.e. a sequence of two contiguous line segments with slopes that have opposite signs. There are also instances where the time-series data is multi-dimensional (say $(x(t), y(t))$), and the user may be interested in knowing if a "pulse" shape in $x(t)$ is followed by an "exponential decay" shape in $y(t)$.

We propose *shape expressions*, a novel declarative language for specifying sophisticated temporal patterns over (possibly multi-dimensional) time series. A shape expression is in essence a regular expression where atomic predicates are arbitrary (linear, exponential, sinusoidal, etc.) shapes with (slope, offset, frequency, etc.) parameters, and with additional parameter constraints. We associate to shape expressions a *noisy language* that allows observed data to approximately match the expression. The noisy expression semantics combines classical regular expression semantics with statistical regression, which is used to match atomic shapes and infer parameter valuations that minimize the noise between the ideal shape and the observation. We allow either using *mean squared error* (MSE) or the *coefficient of determination* (CoD), statistical measures of how close the observed data are to the fitted regression (atomic) shape, as our noise metric. We define *shape automata* as an executable formalism for matching shape expressions and propose a heuristic for querying time series with shape expressions efficiently. We apply this algorithm to two case studies from different CPS and IoT domains to demonstrate its applicability.

Illustrating Example We use the example depicted in Figure 1 to illustrate the concepts presented in this paper. This figure shows a raw noisy signal that contains two pulses. The two pulses differ both in duration, depth and offset, but have the same qualitative shape that characterizes them as pulses. Fig. 1b shows a specification of an ideal pulse. We characterize a pulse as a sequence of 5 segments: (1) constant segment at some b; (2) linearly decreasing segment with slope $a_2 < 0$; (3) constant segment at some b_3; (4) linearly increasing segment with slope $a_4 > 0$; and (5) constant segment at b. We observe that the above specification uses parametric shapes, where the parameters are possibly constrained (e.g. $a_2 < 0$) or shared between shapes (e.g. b), and describes a perfect shape without accounting for noise.



(a) Two pulse shapes

Fig. 1

(b) Idealized Pulse shape

Related Work Regular expressions and temporal logics are the most common general purpose specification languages for expressing temporal patterns in the formal methods community. However, specifying temporal patterns in data is a problem that has been pervasively studied. For instance, specification and recognition of a pulse in pulse-based communications is an IEEE standard [1] in its own right. Extracting unspecified motifs in time series has been studied in data-mining [21], and feature extraction using patterns has been studied in machine learning [20, 12]. More recently, time series shapelets were introduced in [29] as a data mining primitive. A shapelet is a time series segment representing a certain shape identified from data. Our work is partially motivated by the concept of shapelets. In contrast to shapelets that are extracted from unlabelled data, shape expressions provide a more supervised feature extraction mechanism, in which domain-specific knowledge is used to express shapes of interest.

In the context of CPS, timed regular expressions (TRE) [7, 6], quantitative regular expressions (QRE) [3, 2, 4, 19], Signal Temporal Logic (STL) [18] and various stream languages [10, 11, 17, 15, 16] have been used as popular formalisms for specifying properties of CPS behaviors. QREs is a powerful formalism that combines quantitative computations over data with regular expression-based matching. An offline algorithm for matching TREs was proposed in [23, 22]. This thread of work was extended to online pattern matching in [24]. Automata-based matching for TREs has been developed in [25–27]. In contrast to our approach, pattern matching with QREs and TREs is sensitive to noise in data. The problem of uncertainty has been studied through parameterized TRE specifications, either by having parameters in time bounds [5] or in spatial atomic predicates [8]. These approaches are orthogonal to ours – instead of having parameters on standard TRE operators, we focus on a rich class of parameterized atomic shapes. Finally, a sophisticated algorithm to incrementally detect exponential decay patterns in CO_2 measurements was proposed in [28] in the context of smart building applications. We adapt and extend this basic idea to a general purpose specification language that allows combining such atomic shapes with regular operators.

2 Shape Expressions and Automata

In this section, we define *shape expressions* as our pattern specification language. In essence, they are regular expressions over parametrized signal shapes, such as linear, exponential or sine segments, and with additional parameter constraints. We then define *shape automata*, which translate shape expressions and provide an executable formalism for recognizing composite signals made of several types of segments. This executable formalism captures exactly the notion of shape expression, and will allow us to define a family of pattern matching algorithms as we will see in Section 3. We first give a few basic definitions necessary to our framework, such as notions of *signals*, *parameters*, and *shapes*.

2.1 Definitions

Let $P = \{p_1, \ldots, p_n\}$ be a set of *parameters*. A parameter valuation v maps parameters $p \in P$ to values $v(p) \in \mathbb{R} \cup \{\bot\}$, where $\bot$ represents the *undefined*

value. We use the shortcut $v(P)$ to denote $\{v(p_1), \ldots, v(p_n)\}$. A *constraint* γ over P is a Boolean combination of inequalities over P. We write $v \models \gamma$ when the constraint γ is satisfied by the valuation v. Given $p \in P$ and $p \circ k$ for $\circ \in \{=, <, \leq, >, \geq\}$ and some $k \in \mathbb{R}$, we have that $v(p) = \perp$ implies that $v \not\models p \circ k$. We denote by $\Gamma(P)$ the set of all constraints over P.

Let X be a set of signal variables. A *signal* w over X is a function $w : X \times [0, d) \to \mathbb{R}$, where $[0, d)$ is the time domain of w, which we assume to be discrete, hence a subset of $\mathbb{Z}$. We denote by $|w| = d$ the length of w.

Given two signals $w_1 : X \times [0, d_1) \to \mathbb{R}$ and $w_2 : X \times [0, d_2) \to \mathbb{R}$, we denote by $w \equiv w_1 \cdot w_2$ their concatenation $w : X \times [0, d_1 + d_2) \to \mathbb{R}$, where for all $x \in X$, $w(x, t) = w_1(x, t)$ if $t \in [0, d_1)$ and $w(x, t) = w_2(x, t - d_1)$ if $t \in [d_1, d_1 + d_2)$. Let $w : X \times [0, d) \to \mathbb{R}$ be a signal, and d_1 and d_2 be two constants such that $0 \leq d_1 < d_2 \leq d$. We denote by $w^{[d_1, d_2)} : X \times [0, d_2 - d_1) \to \mathbb{R}$ the restriction of w to the time domain $[d_1, d_2)$, such that for all $x \in X$ and $t \in [0, d_2 - d_1)$, $w^{[d_1, d_2)}(x, t) = w(x, t + d_1)$. We allow signals of null duration $d = 0$, which results in the unique signal with the empty time domain[1].

Consider two sequences $\mathbf{y} = y_1, \ldots, y_n$ and $\mathbf{f} = f_1, \ldots, f_n$ of values, where $\mathbf{y}$ represents a sequence of observations and $\mathbf{f}$ the corresponding sequence of predictions given by a model which approximates the distribution of $\mathbf{y}$. The *mean squared error* $\mathrm{MSE}(\mathbf{y}, \mathbf{f})$ of $\mathbf{f}$ relative to $\mathbf{y}$ is a statistical measure of how well the predictions of a (regression) model approximates the observations, and is defined as follows.

$$\mathrm{MSE}(\mathbf{y}, \mathbf{f}) = \frac{1}{n} \Sigma_{i=1}^{n} (y_i - f_i)^2$$

Another statistical measure in a regression analysis of how well the predictions of a (regression) model approximates the observations is the *coefficient of determination* R^2, defined in terms of the *mean* $\bar{y}$ of the sequence $\mathbf{y}$, its total sum of squares SS_{tot} and the residual sum of squares SS_{res} as follows:

$$R^2(\mathbf{y}, \mathbf{f}) = 1 - \frac{SS_{res}(\mathbf{y}, \mathbf{f})}{SS_{tot}(\mathbf{y})} \qquad \bar{y} = \frac{1}{n} \Sigma_{i=1}^{n} y_i$$
$$SS_{tot}(\mathbf{y}) = \Sigma_{i=1}^{n} (y_i - \bar{y})^2 \quad SS_{res}(\mathbf{y}, \mathbf{f}) = \Sigma_{i=1}^{n} (y_i - f_i)^2$$

The coefficient of determination R^2 typically ranges from 0 to 1. An R^2 of 1 indicates that the predictions are a perfect match of the observations. On the contrary, an R^2 of 0 indicates that the model explains none of the variability of the response data around its mean. Negative values of R^2 can occur if the predictions fit the observations worse than a horizontal hyperplane.

2.2 Shape Expressions

We now define the syntax and semantics of *shape expressions* defined over the set X of signals and the set P of parameter variables. A *shape* $\sigma_x(P')$ is an expression that maps parameter variables $P' \subseteq P$ and the signal variable $x \in X$ to a parameterized family of idealized signals. To every shape σ_x, we associate a

[1] The signal with the empty time domain is equivalent to the empty word in the classical language theory

special *duration* variable $\underline{l}_{\sigma,x}$ that is included in the set P of parameter variables.[2] Consider the basic shapes below.

$$\mathsf{lin}_x(a,b,\underline{l}) \equiv \{w \mid \exists v.|w| = v(\underline{l}) \wedge w(x,t) = t \cdot v(a) + v(b)\} \quad (1)$$

$$\mathsf{exp}_x(a,b,c,\underline{l}) \equiv \{w \mid \exists v.|w| = v(\underline{l}) \wedge w(x,t) = v(a) + v(b)e^{t \cdot v(c)}\} \quad (2)$$

$$\mathsf{sin}_x(a,b,c,d,\underline{l}) \equiv \{w \mid \exists v.|w| = v(\underline{l}) \wedge w(x,t) = v(a) + v(b)\sin(v(c)t + v(d))\} \quad (3)$$

In (1), we describe a line segment parameterized by its slope a, and intercept b. In (2), we describe an exponential shape with parameters a, b, c, and $\underline{l}$, while (3) describes a parameterized family of sinusoidal shapes with the specified parameters[3]. Given a valuation v and a shape $\sigma_x(P')$, we denote by $w(x) = \sigma_x(v(P'))$ the signal w that instantiates the shape σ_x to concrete parameter values defined by v. We assume a finite set Σ of shapes, without imposing further restrictions. Shape expressions (SE) are regular expressions, where shapes with unknown parameters play the role of atomic primitives, and which have an additional restriction operator for enforcing parameter constraints.

Definition 1 (SE syntax). *The shape expressions are given by the grammar*

$$\varphi ::= \epsilon \mid \sigma_x(P') \mid \varphi_1 \cup \varphi_2 \mid \varphi_1 \cdot \varphi_2 \mid \varphi^* \mid \varphi : \gamma$$

where $\sigma \in \Sigma$, $x \in X$, $P' \subseteq P$, and $\gamma \in \Gamma(P)$.

The symbol ϵ denotes the *empty word*, the operators $\varphi_1 \cup \varphi_2$, $\varphi_1 \cdot \varphi_2$ and φ^* denote the classical regular expression *union*, *concatenation* and *Kleene star* respectively, while $\varphi : \gamma$ says that φ is *constrained* by γ. We write φ^i as an abbreviation of $\varphi \cdots \varphi$ (i times). We denote by $\Sigma_X(P)$ the set of expressions of the form $\sigma_x(P')$ for $\sigma \in \Sigma$, $x \in X$ and $P' \subseteq P$. The set of shape expressions over P and X is denoted $\Phi(P,X)$.

Example 1. Consider the visual pulse specification from Figure 1b. We describe an ideal pulse as a shape expression φ_{pulse} as follows[4]:

$$\varphi \equiv \mathsf{lin}_x(0,b) \cdot \mathsf{lin}_x(a_2,b_2) : a_2 < 0 \cdot \mathsf{lin}_x(0,b_3) \cdot \mathsf{lin}_x(a_4,b_4) : a_4 > 0 \cdot \mathsf{lin}_x(0,b)$$

The semantics of shape expressions is given as a relation between signals and parameter valuations, which we call a *language*. We associate with every shape expression a *noisy language* $\mathcal{L}_\nu$ for some noise tolerance threshold $\nu \geq 0$, capturing the ν-approximate meaning of the expression. The *exact language* $\mathcal{L}$ capturing the precise meaning of the expression is obtained by setting ν to zero.

[2] We use $\underline{l}$ instead of $\underline{l}_{\sigma,x}$ whenever its association to σ_x is clear from the context, and omit $\underline{l}_{\sigma,x}$ altogether when not interested in the duration of the shape.

[3] We omit the duration variable $\underline{l}$ whenever we are not interested in the duration of a shape - for instance we then use the notation $\mathsf{sin}(a,b,c,d)$.

[4] We abuse the notation and replace a parameter variable by a constant, for instance $\mathsf{lin}_x(0,b)$, as a shortcut for $\mathsf{lin}_x(a_1,b) \ : \ a_1 = 0$.

To define the noisy language of an expression, we associate a goodness of fit measure of a signal to an ideal shape, describing how far is the observed signal from the ideal shape. We derive this measure by combining mean squared error (MSE) computed on atomic shapes. The overall measure gives the quality of a match to a shape expression. We formally define the noisy language as follows.

Definition 2 (SE noisy language). *Let $\nu \in \mathbb{R}_{\geq 0}$ be a noise tolerance threshold. The noisy language $\mathcal{L}_\nu$ of a shape expression is defined as follows:*

$$
\begin{aligned}
\mathcal{L}_\nu(\epsilon) &= \{(w, v) \mid |w| = 0\} \\
\mathcal{L}_\nu(\sigma_x(P')) &= \{(w, v) \mid |w| = v(\underline{l}) \text{ and } \mu(w(x), \sigma_x(v(P'))) \leq \nu\} \\
\mathcal{L}_\nu(\varphi_1 \cdot \varphi_2) &= \{(w_1 \cdot w_2, v) \mid (w_1, v) \in \mathcal{L}_\nu(\varphi_1) \text{ and } (w_2, v) \in \mathcal{L}_\nu(\varphi_2)\} \\
\mathcal{L}_\nu(\varphi_1 \cup \varphi_2) &= \mathcal{L}_\nu(\varphi_1) \cup \mathcal{L}_\nu(\varphi_2) \\
\mathcal{L}_\nu(\varphi^*) &= \bigcup_{i=0}^{\infty} \mathcal{L}_\nu(\varphi^i) \\
\mathcal{L}_\nu(\varphi : \gamma) &= \{(w, v) \mid (w, v) \in \mathcal{L}_\nu(\varphi) \text{ and } v \models \gamma\}
\end{aligned}
$$

where $\mu(\boldsymbol{y}, \boldsymbol{f})$ is substituted by either $\mathrm{MSE}(\boldsymbol{y}, \boldsymbol{f})$ *or* $1 - \mathrm{CoD}(\boldsymbol{y}, \boldsymbol{f})$.

The noisy SE language is defined as the set of all signal/parameter valuation pairs, such that the distance of the signal from the ideal shape signal defined by the shape expression and instantiated by the parameter valuation is smaller or equal than the noise threshold.

Example 2. Consider the shape expression φ_{pulse} specifying a pulse, the signal w depicted in Figure 1a, and the signal $w' = w^I$ the restriction of w to the interval $I = [7, 26)$. Let us consider $v = (v(a_2), v(a_4), v(b), v(b_2), v(b_3), v(b_4)) = (-0.67, 0.67, 9, 17, 7, -5)$ the valuation of parameter variables in φ_{pulse} that instantiates the ideal shape (red line) of the first pulse depicted in Figure 1a. Let $w_1 = w^{[7,12)}$, $w_2 = w^{12,15)}$, $w_3 = w^{[15,18)}$, $w_4 = w^{[18,21)}$ and $w_5 = w^{[21,26)}$, with:

$$
\begin{array}{ll}
\mathrm{MSE}(w_1(x), \mathsf{lin}_x(0, v(b))) = 0.04 & \mathrm{MSE}(w_4(x), \mathsf{lin}_x(v(a_4), v(b_4))) = 0.35 \\
\mathrm{MSE}(w_2(x), \mathsf{lin}_x(v(a_2), v(b_2))) = 0.49 & \mathrm{MSE}(w_5(x), \mathsf{lin}_x(0, v(b))) = 0.10 \\
\mathrm{MSE}(w_3(x), \mathsf{lin}_x(0, v(b_3))) = 0.13 &
\end{array}
$$

It follows that $(w', v) \in \mathcal{L}_{0.5}(\varphi_{pulse})$ but $(w', v) \notin \mathcal{L}_{0.1}(\varphi_{pulse})$.

2.3 Shape Automata

We now define *shape automata*, which will act as recognizers for shape expressions. They are akin to finite state automata in which edges are labeled by shape expressions with unknown parameters, and parameter constraints. We will then show that they are inter-translatable to shape expressions.

Definition 3 (Shape automata). *A shape automaton is a tuple $\langle P, X, Q, \Delta, S, F \rangle$, where (1) P is the set of* parameters*, (2) X is the set of real-valued* signal variables*, (3) Q is the set of control* locations*, (4) $\Delta \subseteq Q \times \Sigma_X(P) \times \Gamma(P) \times Q$ is*

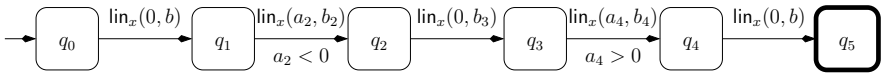


Fig. 2: Shape automaton $\mathcal{A}_{pulse}$

the set of edges*, (5) $S \subseteq Q$ is the set of* starting *locations, and (6) $F \subseteq Q$ is the set of* final *locations.*

Example 3. The shape automaton $\mathcal{A}_{pulse}$, shown in Figure 2 recognizes pulse shapes specified by the shape expression φ_{pulse}.

A state in a shape automaton is a pair (q, v) where q is a location and v is a parameter valuation. The runs of shape automata are akin to those in weighted automata and defined as follows. For a signal w we define transitions $\xrightarrow[c]{w}$ between two states as follows. We have $(q, v) \xrightarrow[c]{w} (q', v')$ if there exists $(q, \sigma_x(P'), \gamma, q') \in \Delta$ such that $P' \subseteq P$, $c = \mu(w(x), \sigma_x(v'(P')))$, $v' \models \gamma$, $v'(p) = v(p)$ for all $p \in P \backslash P'$ and $v'(p) = v(p)$ also for all $p \in P \cap P'$ such that $v(p) \neq \bot$. The semantics of a shape automaton are given as follows.

Definition 4 (Shape automaton run). *A run of a shape automaton over some signal w is a sequence of transitions*

$$(q_0, v_0) \xrightarrow[c_1]{w_1} (q_1, v_1) \xrightarrow[c_2]{w_2} \dots \xrightarrow[c_n]{w_n} (q_n, v_n)$$

such that $q_0 \in S$, $v_0 = (\bot, \dots, \bot)$ and $q_n \in F$, where $w_1 \cdot w_2 \dots w_n$ is a decomposition of w. Such a run ρ induces $\mathrm{cost}(\rho) = \max_{i=1}^{n} c_i$ *and the parameter valuation* $\mathsf{val}(\rho) = v_n$.

The set of runs of a shape automaton $\mathcal{A}$ over some signal w is denoted $\mathcal{R}(\mathcal{A}, w)$. A shape automaton $\mathcal{A}$ associates any given signal w to a similarity measure that is the minimum among the similarity measures of all runs.

Definition 5 (SA language and noisy language). *The noisy language of a shape automaton for a given noise tolerance threshold $\nu \in \mathbb{R}_+$ is $\mathcal{L}_\nu(\mathcal{A}) = \{(w, v) \mid \exists \rho \in \mathcal{R}(\mathcal{A}, w)$ s.t. $\mathsf{val}(\rho) = v$ and $\mathrm{cost}(\rho) \leq \nu\}$. The exact language of a shape automaton is $\mathcal{L}(\mathcal{A}) = \mathcal{L}_0(\mathcal{A})$.*

Example 4. Consider the signal $w' = w_1 w_2 w_3 w_4 w_5$ from Example 2 and let:

$$\begin{array}{lll} v_1 = (\bot, \bot, 9, \bot, \bot, \bot) & c_1 = 0.04 & v_4 = (-0.67, 0.67, 9, 17, 7, -5)\ c_4 = 0.35 \\ v_2 = (-0.67, \bot, 9, 17, \bot, \bot) & c_2 = 0.49 & v_5 = (-0.67, 0.67, 9, 17, 7, -5)\ c_5 = 0.10 \\ v_3 = (-0.67, \bot, 9, \bot, 7, \bot) & c_3 = 0.13 & \end{array}$$

We then have, assuming $v_0 = (\bot, \bot, \bot, \bot, \bot, \bot)$, that

$$\rho = (q_0, v_0) \xrightarrow[c_1]{w_1} (q_1, v_1) \xrightarrow[c_2]{w_2} \cdots \xrightarrow[c_5]{w_5} (q_5, v_5)$$

is a run of $\mathcal{A}_{pulse}$ over w' with $\mathrm{cost}(\rho) = 0.49$ and $w' \in \mathcal{L}_{0.5}(\mathcal{A}_{pulse})$.

We now formally show the equivalence between shape expressions and shape automata. The first direction of the theorem allows to construct automata recognizers for arbitrary expressions. The second direction of the theorem shows that shape expressions are expressively complete relative to the class of automata under consideration.

Theorem 1 (SE $\Leftrightarrow$ SA). *For any shape expression φ there exists a shape automaton $\mathcal{A}_\varphi$ such that $\mathcal{L}_\nu(\mathcal{A}_\varphi) = \mathcal{L}_\nu(\varphi)$ for all $\nu \geq 0$. For any shape automaton $\mathcal{A}$ there exists a shape expression $\varphi_\mathcal{A}$ such that $\mathcal{L}_\nu(\varphi_\mathcal{A}) = \mathcal{L}_\nu(\mathcal{A})$ for all $\nu \geq 0$.*

3 Pattern Matching

In Section 2.3, we introduced shape automata to recognize signals that are close to a specified shape. However, a shape expression is not intended to represent a whole signal, but only a segment thereof. In this section, we extend shape automata to enable them identifying all signal segments that match specific shapes. We first define the notion of noisy match sets.

Definition 6 (Noisy match set). *For any signal w defined over a time domain $\mathbb{T} = [0, d)$, shape expression φ and noise tolerance threshold ν, we define the* match set $\mathcal{M}(\varphi, w)$ *and the* noisy match set $\mathcal{M}_\nu(\varphi, w)$ *as follows:*

$$\mathcal{M}_\nu(\varphi, w) = \{(t, t') \in \mathbb{T}^2 \mid t \leq t' \text{ and } w^{[t,t')} \in \mathcal{L}_\nu(\varphi)\}$$

Given a shape automaton $\mathcal{A}$, its associated *shape pattern matching automaton* $\hat{\mathcal{A}}$ is another shape automaton that extends $\mathcal{A}$ with dedicated initial and final locations, which allow $\hat{\mathcal{A}}$ to silently consume a prefix and a suffix of a signal. The construction follows [9] and is given in the definition below.

Definition 7 (Shape pattern matching automaton). *Let $\mathcal{A} = \langle P, X, Q, \Delta, S, F\rangle$ be a shape automaton. Then the corresponding* shape pattern matching automaton *is $\hat{\mathcal{A}} = \langle P, X, \hat{Q}, \hat{\Delta}, \hat{S}, \hat{F}\rangle$, where*

- $\hat{Q} = Q \cup \{\hat{s}, \hat{f}\}$, $\hat{S} = \{\hat{s}\}$, $\hat{F} = \{\hat{f}\}$,
- $\hat{\Delta} = \Delta \cup \{(\hat{s}, \mathsf{any}, \mathsf{true}, q) \mid q \in S\} \cup \{(q, \mathsf{any}, \mathsf{true}, \hat{f}) \mid q \in F\}$, *where* any *is a special shape such that $\mu(w, \mathsf{any}) = 0$ for all w.*

Intuitively, given a signal w, a shape expression φ and its associated shape pattern matching automaton $\hat{\mathcal{A}}_\varphi$, an accepting run ρ over w decomposed into $w_0 \cdot w_1 \cdots w_{n+1}$ in $\hat{\mathcal{A}}_\varphi$

$$(\hat{s}, v_0) \xrightarrow[0]{w_0} (q_0, v_0) \xrightarrow[c_1]{w_1} \dots \xrightarrow[c_n]{w_n} (q_n, v_n) \xrightarrow[0]{w_{n+1}} (\hat{f}, v_n)$$

represents one potential match (defined by segment (t, t') in w where $t = |w_0|$ and $t' = |w| - |w_{n+1}|$) with one specific parameter instantiation (v_n) and its associated similarity measure $\text{cost}(\rho) = \max_{i=1}^{n} c_i$. We denote by $\lambda(\rho) = (t, t')$ the *label* of run ρ over w in $\hat{\mathcal{A}}$. We first note that for a given decomposition of w,

there is an infinite number of runs over w in $\hat{\mathcal{A}}_\varphi$ that follow that decomposition due to the parameters being valued as real numbers. We also note that for a given signal w, there is a finite (but large) number of its decompositions.

Example 5. Figure 3 shows three runs ρ_1, ρ_2 and ρ_3 over w in $\hat{\mathcal{A}}_{pulse}$ and the corresponding ideal shapes defined by the valuations computed during the runs. We can see that each run identifies one segment of w that could be a potential match of the shape expression φ_{pulse} with specific parameter values and cost. In particular, we can observe that runs ρ_1 and ρ_2 decompose w in the same manner but with different parameter valuations, resulting in $\text{cost}(\rho_1) < \text{cost}(\rho_2)$.

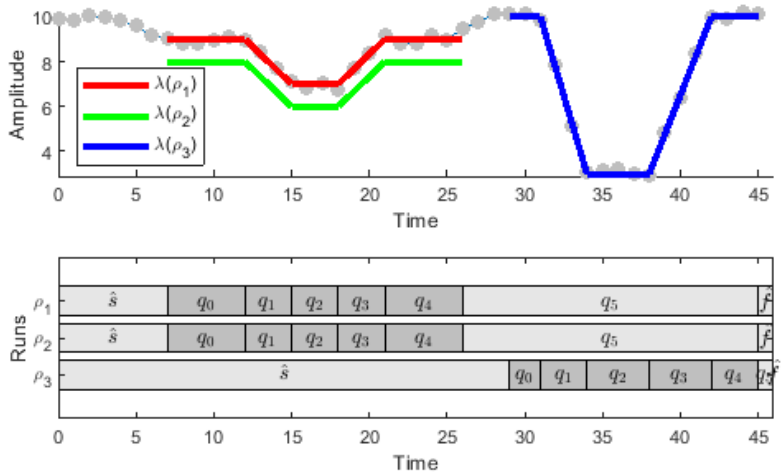


Fig. 3: Pulse train - three runs ρ_1, ρ_2 and ρ_3 over w in $\hat{\mathcal{A}}_{pulse}$.

From the above observations, we obtain that the labeling of the set of runs associated to a shape pattern matching automaton $\hat{\mathcal{A}}$ and a signal w gives us exactly the match set of $\mathcal{L}(\mathcal{A})$ relative to w.

Theorem 2. *Let φ be a shape expression, $\hat{\mathcal{A}}_\varphi$ the corresponding shape pattern matching automaton, w a signal and ν a noise tolerance threshold. We have that $\mathcal{M}_\nu(\varphi, w) = \{(t, t') \mid \exists \rho \in \mathcal{R}(\hat{\mathcal{A}}_\varphi, w)$ s.t. $\lambda(\rho) = (t, t')$ and $\text{cost}(\rho) \leq \nu\}$.*

We observe that while this in principle solves the SE pattern-matching problem, the complexity in terms of signal length is not practical. Let us define the dot-depth of some expression φ the maximal number of concatenations featured on any branch of its syntax tree.

Theorem 3. *The size of the set of runs of a shape matching automaton $\hat{\mathcal{A}}_\varphi$ is $\Omega(n^{k+2})$, where n is the size of the trace, and k is the dot-depth of φ.*

The dot-depth of any expression is nonnegative, hence this lower bound is at least quadratic in the length of the signal. This means that any exhaustive algorithm

will not scale in many practical applications, where typical signal can be over 10^6 samples long.

We propose two ways to handle complexity: (1) bound the length of matches, or (2) develop heuristics to efficiently match shape expressions. Bounding the length of matches is reflected in the following definition.

Definition 8 (Bounded shape expressions). *A shape expression is said to be* bounded *(*by k*) when for all words w we have that $w \in \mathcal{L}(\varphi)$ implies $|w| \leq k$.*

Theorem 4 (Linear-time upper bound). *For an expression φ bounded by k the size of the set of accepting runs of the shape matching automaton can be represented by a dag of size $O(nk2^{m \cdot k^m})$, where n is the length of the trace and m is the length of the expression.*

4 Policy Scheduler for Shape Matching Automata

In this section, we propose a heuristic in the form of a policy scheduler that efficiently approximates the complete match set by computing a representative subset of non-overlapping matches.

Let w be a signal defined over X and $\sigma_x(P')$ a shape with $x \in X$. We denote by reg the *statistical regression* with constraints which returns the pair of the parameter values $v(P')$ which minimizes MSE under the constraint γ and the associated $\mu(w, \sigma_x(v(P')))$, defined as follows:

$$\mathsf{reg}(w, \sigma_x, \gamma) = (\text{argmin}_v\{\text{MSE}(w, \sigma_x(v(P'))) \mid v \models \gamma\}, \mu(w, \sigma_x(v(P')))) \,.$$

We now show that μ (MSE and CoD) can be computed in an online fashion. Given the two sequences $\mathbf{y} = y_1, \ldots, y_n$ and $\mathbf{f} = f_1, \ldots, f_n$ of observations and predictions, we define a recursive definition of MSE and CoD as follows.

$$\begin{array}{ll}
\text{MSE}(\mathbf{y}, \mathbf{f}, n+1) & = \frac{n}{n+1}\,\text{MSE}(\mathbf{y}, \mathbf{f}, n) + \frac{1}{n+1}(y_{n+1} - f_{n+1})^2 \\
\bar{y}(n+1) & = \frac{n}{n+1}\bar{y}(n) + \frac{1}{n+1}y_{n+1} \\
SS_{tot}(\mathbf{y}, n+1) & = SS_{tot}(\mathbf{y}, n) + (y_{n+1} - \bar{y}(n))(y_{n+1} - \bar{y}(n+1)) \\
SS_{res}(\mathbf{y}, \mathbf{f}, n+1) & = SS_{res}(\mathbf{y}, \mathbf{f}, n) + (y_{n+1} - f_{n+1})^2 \\
R^2(\mathbf{y}, \mathbf{f}, n+1) & = 1 - \frac{SS_{res}(\mathbf{y}, \mathbf{f}, n+1)}{SS_{tot}(\mathbf{y}, n+1)}
\end{array}$$

We require a minimum length $\lambda > 1$ for atomic shape matches[5]. We define the auxiliary method out_Δ as follows:

$$\mathsf{out}_\Delta(S) = \{\delta \mid \exists\, \delta = (q, \sigma_x, \gamma, q') \in \Delta \text{ for some } q \in S\}$$

The method policy_scheduler searches for non-overlapping SE matches in w from time 0, using method expression_match. The call of expression_match at time t returns another time t'. If $t' > t$, the segment $[t, t']$ successfully matches the

[5] We also assume that the SMA $\hat{\mathcal{A}}$, the signal w, the noise tolerance threshold ν and the minimum match length λ are given as global parameters to the main procedure policy_scheduler and are implicitly propagated to all the other methods

Algorithm 1: Shape expression match expression_match

Input: Set of locations S, current end match time t
Output: New end match time t'

```
1 t' ← −∞
2 if S ∩ F ≠ ∅ then t' ← t
3 else if t < |w| then
4     foreach δ = (q, σ_x, γ, q') ∈ out_Δ(S) do
5         τ ← atomic_match(δ, t)
6         if τ > −∞ then τ' ← expression_match({q'}, τ)
7         t' ← max{t', τ'}
8 return t'
```

expression. The segment $[t, t']$ is added to the set of matches and the procedure expression_match is invoked again at time $t' + 1$. If $t' \leq t$, it means that the expression could not be matched from time t. The procedure expression_match is invoked again at time $t + 1$.

The shape matching procedure expression_match (see Algorithm 1) attempts in a recursive fashion to reach a final location from a set of locations S and time index t. The procedure returns another time index t', where $t' \geq t$ if a final location can be reached in $t' - t$ steps from a location in S, or $t' = -\infty$ (the initial value of t', see line 1) otherwise. If one of the locations is a final location, we have that $t' = t$ (lines 2). If none of the locations in S is final, and we have not yet reached the end of w (lines $3-7$), the procedure does the following. For every transition with a source location in S, labeled by σ_x and γ (lines $4-7$), atomic_match computes the end time τ of the longest match of σ_x that satisfies γ and starts at t (line 5). If there is no such match, τ equals to $-\infty$, otherwise $\tau \geq t + \lambda$[6]. For all the transitions that result in a match ending at time τ, we recursively call expression_match with the target location q' and time τ as inputs, and τ' as output (line 6). The procedure keeps the longest from the successful expression matches (line 7). This effectively allows the procedure to concurrently follow multiple paths and select the one that provides the longest match.

The atomic shape matching procedure atomic_match, shown in Algorithm 2, efficiently computes the longest match of an atomic shape starting from a given time index. It takes as inputs a transition $\delta = (q, \sigma_x, \gamma, q')$ and the time index t, and returns the end time t' of the longest σ_x ν-noisy match $[t, t']$ that satisfies γ. The algorithm starts by fitting the shape σ_x to the segment $w' = w^{[t,t+\tau)}$ under the constraint γ, using the regression method reg, and thus estimating the parameters v (lines 3). The procedure reg also returns the corresponding μ-value c of the performed regression. If the associated μ-value c is greater than the allowed noise tolerance ν, the procedure returns $t' = -\infty$, meaning that the segment is not a good candidate for matching the shape. Otherwise, the algorithm iteratively extends the size τ of the segment as long as the μ-value between the extended prefix and $\sigma_x(v(P'))$ instantiated with the fixed parameter

[6] Recall that we require atomic matches of minimum length λ.

Algorithm 2: Atomic shape match atomic_match.

Input: Transition $\delta = (q, \sigma_x, \gamma, q')$, start match time index t
Output: End match time t'

1 $t' \leftarrow -\infty$
2 **if** $t + \lambda \leq |w|$ **then**
3 $\quad \tau \leftarrow \lambda$; $w' \leftarrow w^{[t,t+\tau)}$; $(v, c) \leftarrow \mathsf{reg}(w', \sigma_x(P'), \gamma)$
4 $\quad$ **while** $c \leq \nu$ **do**
5 $\quad\quad t' \leftarrow t + \tau$
6 $\quad\quad$ **if** $t' < |w|$ **then**
7 $\quad\quad\quad \tau \leftarrow \tau + 1$; $w' \leftarrow w' \cdot w(t')$
8 $\quad\quad\quad c \leftarrow \mu(w', \sigma_x(v(P')))$
9 $\quad\quad\quad$ **if** $c > \nu$ **then** $(v, c) \leftarrow \mathsf{reg}(w', \sigma_x(P'), \gamma)$
10 $\quad\quad\quad$ **else break**
11 **return** t'

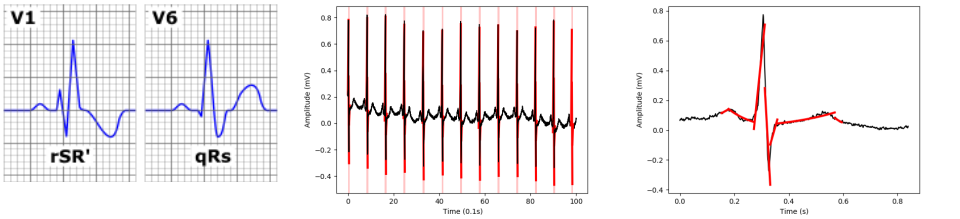
valuation v remains lower than or equal to ν (lines $4 - 10$). We note that each extension of the signal prefix updates μ but not the parameter valuation. There are two possible reasons for μ becoming greater than ν: (i) either the estimated parameter valuation v needs to be updated, or (ii) the current prefix does not fit the shape under the constraint ν anymore with any valuation v. In the first case, the procedure re-estimates the new parameter valuation and re-computes μ (line 9). If the re-computed μ is smaller than or equal to ν and we didn't reach the end of the signal, we repeat the match extension procedure. Otherwise, we terminate the procedure and return the time index t' where the current match (if any, otherwise t' equals to $-\infty$) ended.

5 Implementation and Evaluation

We implemented the Algo. 2 into a prototype tool using the Python programming language. We employed pattern matching of shape expressions to two applications – detection of patterns in electro-cardiograms (ECG) and oscillatory behaviors in an aircraft elevator control system. All experiments were run on MacBook Pro with the Intel Core i7 2.6 GHz processor and 16GB RAM.

5.1 Detection of Anomalous Patterns in ECG

In this case study, we consider ECG signals from the PhysioBank database [14], which contains 549 records from 290 subjects (209 male and 81 female, aged from 17 to 87). Each record includes 15 simultaneously measured signals, digitized at 1,000 samples per second, with 16-bit resolution over a range of $\pm 16.384 mV$. The diagnostic classes for the subjects participating in the recordings include cardiovascular diseases such as myocardial infarction, cardiomyopathy, dysrythmia and myocardial hypertrophy.



(a) RBBB characteristics on channels v1, v6

(b) Signal on v6 channel

(c) Magnified anomalous pulse

Fig. 4: Recognizing pulses in ECG signals

Specification of an Anomalous Heart Pulse. We consider the *right bundle branch block* (RBBB) heart condition, in which the right ventricle is not directly activated by impulses traveling through the right bundle branch. Fig. 4a depicts a visual characterization of the RBBB heart condition as it can be observed on channels v1 and v6[7]. In this work, we concentrate on specifying the shape of the pulse depicted in v6 using shape expressions. The specification φ of the anomalous v6 pulse consists of a sequence of 7 atomic shapes:

$$\begin{aligned}\varphi = \mathsf{exp}(a_1, b_1, c_1) \ : \ b_1 > 0 \cdot \mathsf{exp}(a_2, b_2, c_2) \ : \ b_2 < 0 \cdot \\ \mathsf{lin}(a_3, b_3) \ : \ a_3 > 0 \cdot \mathsf{lin}(a_4, b_4) \ : \ a_4 < 0 \cdot \mathsf{lin}(a_5, b_5) \ : \ a_5 > 0 \cdot \\ \mathsf{exp}(a_6, b_6, c_6) \ : \ b_6 > 0 \cdot \mathsf{exp}(a_7, b_7, c_7) \ : \ b_7 < 0\end{aligned}$$

Evaluation. We evaluated our SE matching procedure with respect to the recordings of a 70 year old patient that suffers from RBBB condition. The v6 channel recording of the patient, shown in Figure 4b, has 10,000 samples. In this experiment, we use CoD as our noise metric[8]. With noise threshold $\nu = 0.02$, we were able to identify all the segments that match φ in $28.98s$. The matches are depicted as colored vertical bands in Figure 4b. Figure 4c zooms in on a single match and shows the ideal shape that was inferred to match the pattern.

We now experimentally study how sensitive is the quality of the procedure outcome with respect to the noise threshold and the constraints on the parameters, and how well the procedure scales with the size of the input.

Sensitivity to the noise threshold and the constraints on the parameters. Domain knowledge in a particular application field can be used to derive more precise specifications. In the case of anomalous v6 pulses for patients with RBBB condition, such knowledge can be for instance used to refine its specification φ by further constraining the slope a_3 to be greater than 0.5, resulting in specification φ'. We demonstrate the impact of the noise threshold to the quality of pattern matching in the cases of under-specified (φ) and over-specified (φ') shape expressions. Table 1a shows the results of the experiments, where column $|H|$ denotes

[7] The figure is under copyright by A. Rad.

[8] We recall that $\nu = 0$ denotes zero noise tolerance and $\nu = 1$ allows arbitrary level of noise.

Table 1: Experimental Results

(a) Sensitivity to the noise threshold

ν	$\lvert H\rvert$	$\lvert\mathcal{M}_\nu(\varphi)\rvert$	$\lvert\mathcal{M}_\nu(\varphi')\rvert$
0.70	4	9	4
0.24	4	7	4
0.20	4	5	4
0.10	4	4	4
0.02	4	4	4
0.01	4	0	0

(b) Runtime and memory requirements

Num. Samples	Runtime (s)	Mem. (MB)
1,000	0.46	33.13
2,500	1.43	48.82
5,000	3.39	70.80
7,500	6.39	72.83
10,000	10.12	89.18

the number of segments matched by the inspection of the signal by a human with domain knowledge and columns $|\mathcal{M}_\nu(\varphi)|$ and $|\mathcal{M}_\nu(\varphi')|$ denotes the number of the segments matching the expressions φ and φ' by our procedure, respectively.

We first observe that domain knowledge iimproves the quality of both the specification the robustness of the monitor. Second, our approach can result in missing patterns or detecting false patterns. This result is expected – very low ν enables to only match shapes that are very close to the ideal one, while very high ν results in matching shapes that are far away from the specification. Hence, our procedure may require tuning parameters.

Scalability. We now evaluate the scalability of our procedure with respect to the size of the signal, taking into account the computation time and the memory requirements. Table 1b summarizes the results. The computation time in this experiment exhibits an almost linear behavior, while the memory consumption appears to grow in a sub-linear fashion with respect to the size of the input.

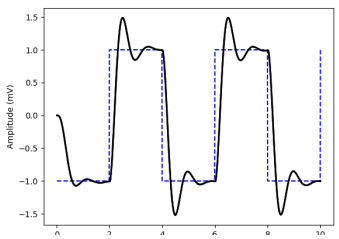
5.2 Detection of Ringing in an Aircraft Elevator Control System

In many electronics applications, step response is used to study how the system responds to sudden changes in inputs. *Ringing* is an oscillation in the output signal, which is encountered in response to a step in input. It is considered to be an undesirable behavior, which nevertheless cannot be fully avoided. It is hence important to investigate properties of the oscillations (amplitude, frequency, etc.) to determine the quality of the output response.

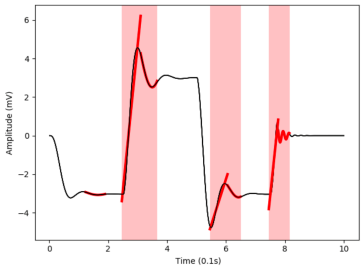
We use SEs to detect and study ringing behavior in an aircraft elevator control system [13]. It is a Simulink model of a redundant actuator control system with one elevator on the left and one on the right side. In essence, the pilot gives a command with the intended position of the aircraft, which must be followed by the left and right elevators. When the pilot gives a step command, this results in the ringing response by the control system, as shown in Figure 5 (a).

Specification of a Ringing Behavior. We are interested in detecting both the rising and falling edge and the subsequent ringing behavior. We chose to specify such behavior as a line, followed by a sinc wave ($sinc(a,b,c,d,t) = a + b\frac{sin(ct+d)}{ct+d}$).

$$\varphi = \mathsf{lin}_x(a_1, b_1) : a_1 > 0.5 \cdot \mathsf{sinc}_x(a_2, b_2, c_2, d_2).$$



(a) Step response of the system.



(b) Segments matching ringing patterns.

Fig. 5: Aircraft Elevator Control System Step Response

Amp	a_1	b_1	a_2	b_2	c_2	d_2
1	1.36	-8.98	-0.40	3.03	-2.05	17.73
2	2.83	-18.55	-1.51	2.83	-3.31	25.80
3	4.75	-30.75	-2.78	-8.76	-5.21	13.09

Table 2: Parameters inferred from segments matching φ.

Inferring Parameters of Ringing Patterns. Fig. 5 (b) shows the segments in the output response of the aircraft elevator control system that match the ringing pattern. We stimulate the system with input steps of different amplitudes and show how this change in inputs affects the step response and the resulting ringing oscillations. For each response signal, we report the inferred parameters in Table 2. We can observe that the rising edge of the step response becomes steeper with input steps of higher amplitude. We can also see that both the amplitude and the frequency of the sinc monotonically decrease with the input amplitude.

6 Conclusion

In this paper, we proposed *shape expressions* as a language for specification of rich and complex temporal patterns. We studied essential properties of shape expressions and developed an efficient heuristic pattern matching procedure for this specification language. We believe that this work explores the expressiveness boundaries of declarative specification languages.

We will pursue this work in several directions. We will apply our technique to examples from more application domains. We will study more sophisticated matching methods that will minimize the need of tuning parameter constraints. We will compare more closely our approach to the work on classical regular expression matching on one hand, and purely machine learning feature extraction methods on the other hand. We will finally investigate the application of shape expressions in testing CPS with the particular focus on generating test cases from such a specification language.

Acknowledgments This research was supported in part by the Austrian Science Fund (FWF) under grants 27 S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award), and by the Productive 4.0 project (ECSEL 737459).

References

1. IEEE standard on pulse ment and analysis by objective techniques. *IEEE Std. 181-1977*, 1977.
2. Houssam Abbas, Alena Rodionova, Ezio Bartocci, Scott A Smolka, and Radu Grosu. Quantitative regular expressions for arrhythmia detection algorithms. In *International Conference on Computational Methods in Systems Biology*, pages 23–39. Springer, 2017.
3. Rajeev Alur, Dana Fisman, and Mukund Raghothaman. Regular programming for quantitative properties of data streams. In *European Symposium on Programming*, pages 15–40. Springer, 2016.
4. Rajeev Alur, Konstantinos Mamouras, and Caleb Stanford. Modular quantitative monitoring. *Proceedings of the ACM on Programming Languages*, 3(POPL):50, 2019.
5. Étienne André, Ichiro Hasuo, and Masaki Waga. Offline timed pattern matching under uncertainty. In *23rd International Conference on Engineering of Complex Computer Systems, ICECCS 2018, Melbourne, Australia, December 12-14, 2018*, pages 10–20, 2018.
6. Eugene Asarin, Paul Caspi, and Oded Maler. A Kleene theorem for timed automata. In *Logic in Computer Science (LICS)*, pages 160–171, 1997.
7. Eugene Asarin, Paul Caspi, and Oded Maler. Timed regular expressions. *Journal of ACM*, 49(2):172–206, 2002.
8. Alexey Bakhirkin, Thomas Ferrère, Oded Maler, and Dogan Ulus. On the quantitative semantics of regular expressions over real-valued signals. In *Formal Modeling and Analysis of Timed Systems - 15th International Conference, FORMATS 2017, Berlin, Germany, September 5-7, 2017, Proceedings*, pages 189–206, 2017.
9. Alexey Bakhirkin, Thomas Ferrère, Dejan Nickovic, Oded Maler, and Eugene Asarin. Online timed pattern matching using automata. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 215–232. Springer, 2018.
10. Ben D'Angelo, Sriram Sankaranarayanan, César Sánchez, Will Robinson, Bernd Finkbeiner, Henny B. Sipma, Sandeep Mehrotra, and Zohar Manna. LOLA: runtime monitoring of synchronous systems. In *12th International Symposium on Temporal Representation and Reasoning (TIME 2005), 23-25 June 2005, Burlington, Vermont, USA*, pages 166–174, 2005.
11. Peter Faymonville, Bernd Finkbeiner, Sebastian Schirmer, and Hazem Torfah. A stream-based specification language for network monitoring. In *Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings*, pages 152–168, 2016.
12. Pierre Geurts. Pattern extraction for time series classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 115–127. Springer, 2001.
13. Jason Ghidella and Pieter Mosterman. Requirements-based testing in aircraft control design. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*, page 5886, 2005.
14. Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

15. Felipe Gorostiaga and César Sánchez. Striver: Stream runtime verification for real-time event-streams. In *Runtime Verification - 18th International Conference, RV 2018, Limassol, Cyprus, November 10-13, 2018, Proceedings*, pages 282–298, 2018.
16. Sylvain Hallé and Raphaël Khoury. Event stream processing with beepbeep 3. In *RV-CuBES 2017. An International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools, September 15, 2017, Seattle, WA, USA*, pages 81–88, 2017.
17. Martin Leucker, César Sánchez, Torben Scheffel, Malte Schmitz, and Alexander Schramm. Tessla: runtime verification of non-synchronized real-time streams. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC 2018, Pau, France, April 09-13, 2018*, pages 1925–1933, 2018.
18. Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings*, pages 152–166, 2004.
19. Konstantinos Mamouras, Mukund Raghothaman, Rajeev Alur, Zachary G Ives, and Sanjeev Khanna. StreamQRE: Modular specification and efficient evaluation of quantitative queries over streaming data. In *ACM SIGPLAN Notices*, volume 52, pages 693–708. ACM, 2017.
20. Robert T Olszewski. Generalized feature extraction for structural pattern recognition in time-series data. Technical report, Carnegie-Mellon Univ. School of Computer Science, 2001.
21. Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270. ACM, 2012.
22. Dogan Ulus. Montre: A tool for monitoring timed regular expressions. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, pages 329–335, 2017.
23. Dogan Ulus, Thomas Ferrère, Eugene Asarin, and Oded Maler. Timed pattern matching. In *Formal Modeling and Analysis of Timed Systems (FORMATS)*, pages 222–236, 2014.
24. Dogan Ulus, Thomas Ferrère, Eugene Asarin, and Oded Maler. Online timed pattern matching using derivatives. In *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, pages 736–751, 2016.
25. Masaki Waga and Ichiro Hasuo. Moore-machine filtering for timed and untimed pattern matching. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 37(11):2649–2660, 2018.
26. Masaki Waga, Ichiro Hasuo, and Kohei Suenaga. Efficient online timed pattern matching by automata-based skipping. In *Formal Modeling and Analysis of Timed Systems - 15th International Conference, FORMATS 2017, Berlin, Germany, September 5-7, 2017, Proceedings*, pages 224–243, 2017.
27. Masaki Waga, Ichiro Hasuo, and Kohei Suenaga. MONAA: A tool for timed pattern matching with automata-based acceleration. In *3rd Workshop on Monitoring and Testing of Cyber-Physical Systems, MT@CPSWeek 2018, Porto, Portugal, April 10, 2018*, pages 14–15, 2018.

28. Florian Wenig, Peter Klanatsky, Christian Heschl, Cristinel Mateis, and Nickovic Dejan. Exponential pattern recognition for deriving air change rates from CO2 data. In *26th IEEE International Symposium on Industrial Electronics, ISIE 2017, Edinburgh, United Kingdom, June 19-21, 2017*, pages 1507–1512, 2017.
29. Lexiang Ye and Eamonn J. Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 947–956, 2009.