






Mining Interpretable Spatio-Temporal Logic Properties for Spatially Distributed Systems

Sara Mohammadinejad¹(✉) , Jyotirmoy V. Deshmukh¹ ,
and Laura Nenzi^{2,3} 

¹ University of Southern California, Los Angeles, USA

{saramoha, jdeshmuk}@usc.edu

² University of Trieste, Trieste, Italy

lnenzi@units.it

³ TU Wien, Vienna, Austria

Abstract. The Internet-of-Things, complex sensor networks, multi-agent cyber-physical systems are all examples of spatially distributed systems that continuously evolve in time. Such systems generate huge amounts of spatio-temporal data, and system designers are often interested in analyzing and discovering structure within the data. There has been considerable interest in learning causal and logical properties of temporal data using logics such as Signal Temporal Logic (STL); however, there is limited work on discovering such relations on *spatio-temporal* data. We propose the first set of algorithms for *unsupervised learning* for spatio-temporal data. Our method does automatic feature extraction from the spatio-temporal data by projecting it onto the parameter space of a *parametric spatio-temporal reach and escape logic* (PSTREL). We propose an agglomerative hierarchical clustering technique that guarantees that each cluster satisfies a distinct STREL formula. We show that our method generates STREL formulas of bounded description complexity using a novel decision-tree approach which generalizes previous unsupervised learning techniques for Signal Temporal Logic. We demonstrate the effectiveness of our approach on case studies from diverse domains such as urban transportation, epidemiology, green infrastructure, and air quality monitoring.

Keywords: Distributed systems • Unsupervised learning • Spatio-temporal data • Interpretability • Spatio-temporal reach and escape logic

1 Introduction

Due to rapid improvements in sensing and communication technologies, embedded systems are now often spatially distributed. Such spatially distributed

J. V. Deshmukh and L. Nenzi—Equal contribution.

© Springer Nature Switzerland AG 2021

Z. Hou and V. Ganesh (Eds.): ATVA 2021, LNCS 12971, pp. 91–107, 2021.

https://doi.org/10.1007/978-3-030-88885-5_7

systems (SDS) consist of heterogeneous components embedded in a specific topological space, whose time-varying behaviors evolve according to complex mutual inter-dependence relations [16]. In the formal methods community, tremendous advances have been achieved for verification and analysis of distributed systems. However, most formal techniques abstract away the specific spatial aspects of distributed systems, which can be of crucial importance in certain applications. For example, consider the problem of developing a bike-sharing system (BSS) in a “sharing economy.” Here, the system consists of a number of bike stations that would use sensors to detect the number of bikes present at a station, and use incentives to let users return bikes to stations that are running low. The bike stations themselves could be arbitrary locations in a city, and the design of an effective BSS would require reasoning about the distance to nearby locations, and the time-varying demand or supply at each location. For instance, the property “there is always a bike and a slot available at distance d from a bike station” depends on the distance of the bike station to its nearby stations. Evaluating whether the BSS functions correctly is a verification problem where the specification is a *spatio-temporal* logic formula. Similarly, consider the problem of coordinating the movements of multiple mobile robots, or a HVAC controller that activates heating or cooling in parts of a building based on occupancy. Given spatio-temporal execution traces of nodes in such systems, we may be interested in analyzing the data to solve several classical formal methods problems such as fault localization, debugging, invariant generation or specification mining. It is increasingly urgent to formulate methods that enable reasoning about spatially-distributed systems in a way that explicitly incorporates their spatial topology.

In this paper, we focus on one specific aspect of spatio-temporal reasoning: mining interpretable logical properties from data in an SDS. We model a SDS as a directed or undirected graph where individual compute nodes are vertices, and edges model either the connection topology or spatial proximity. In the past, analytic models based on partial differential equations (e.g. diffusion equations) [6] have been used to express the spatio-temporal evolution of these systems. While such formalisms are incredibly powerful, they are also quite difficult to interpret. Traditional machine learning (ML) approaches have also been used to uncover the structure of such spatio-temporal systems, but these techniques also suffer from the lack of interpretability. Our proposed method draws on a recently proposed logic known as *Spatio-Temporal Reach and Escape Logic* (STREL) [2]. Recent research on STREL has focused on efficient algorithms for runtime verification and monitoring of STREL specifications [2, 3]. However, there is no existing work on mining STREL specifications.

Mined STREL specifications can be useful in many different contexts in the design of spatially distributed systems; an incomplete list of usage scenarios includes the following applications: (1) Mined STREL formulas can serve as spatio-temporal invariants that are satisfied by the computing nodes, (2) STREL formulas could be used by developers to characterize the properties of a deployed spatially distributed system, which can then be used to monitor any subsequent

updates to the system, (3) Clustering nodes that satisfy similar STREL formulas can help debug possible bottlenecks and violations in communication protocols in such distributed systems.

There is considerable amount of recent work on learning temporal logic formulas from data [8, 11, 14, 15]. In particular, the work in this paper is closest to the work on unsupervised clustering of time-series data using Signal Temporal Logic [11]. In this work, the authors assume that the user provides a Parametric Signal Temporal Logic (PSTL) formula, and the procedure projects given temporal data onto the parameter domain of the PSTL formula. The authors use off-the-shelf clustering techniques to group parameter values and identify STL formulas corresponding to each cluster. There are a few hurdles in applying such an approach to spatio-temporal data. First, in [11], the authors assume a monotonic fragment of PSTL: there is no such fragment identified in the literature for STREL. Second, in [11], the authors assume that clusters in the parameter space can be separated by axis-aligned hyper-boxes. Third, given spatio-temporal data, we can have different choices to impose the edge relation on nodes, which can affect the formula we learn.

To address the shortcomings of previous techniques, we introduce PSTREL, by treating threshold constants in signal predicates, time bounds in temporal operators, and distance bounds in spatial operators as parameters. We then identify a monotonic fragment of PSTREL, and propose a multi-dimensional binary-search based procedure to infer *tight* parameter valuations for the given PSTREL formula. We also explore the space of implied edge relations between spatial nodes, proposing an algorithm to define the most suitable graph. After defining a projection operator that maps a given spatio-temporal signal to parameter values of the given PSTREL formula, we use an agglomerative hierarchical clustering technique to cluster spatial locations into hyperboxes. We improve the method of [11] by introducing a decision-tree based approach to systematically split overlapping hyperbox clusters. The result of our method produces axis-aligned hyperbox clusters that can be compactly described by an STREL formula that has length proportional to the number of parameters in the given PSTREL formula (and independent of the number of clusters). Finally, we give human-interpretable meanings for each cluster. We show the usefulness of our approach considering four benchmarks: COVID-19 data from LA County, Outdoor air quality data, BSS data and movements of the customer in a Food Court.

Running Example: A Bike Sharing System (BSS). To ease the exposition of key ideas in the paper, we use an example of a BSS deployed in the city of Edinburgh, UK. The BSS consists of a number of bike stations, distributed over a geographic area. Each station has a fixed number of bike slots. Users can pick up a bike, use it for a while, and then return it to another station in the area. The data that we analyze are the number of bikes (B) and empty slots (S) at each time step in each bike station. With the advent of electric bikes, BSS have become an important aspect in urban mobility, and such systems make use of embedded devices for diverse purposes such as tracking bike usage, billing, and displaying information about availability to users over apps. Figure 1b shows the

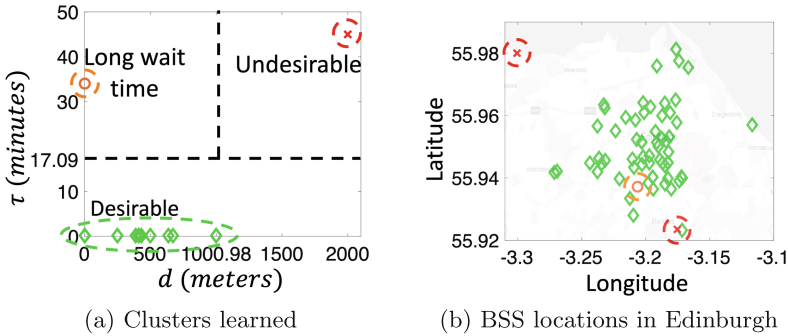


Fig. 1. Interpretable clusters automatically identified by our technique.

map of the Edinburgh city with the bike stations. Different colors of the nodes represent different learned clusters as can be seen in Fig. 1a. For example, using our approach, we learn that stations in *orange* cluster have a long wait time, and stations in *red* cluster are the most undesirable stations as they have long wait time and do not have nearby stations with bike availability. If we look at the actual location of *red* points in Fig. 1b, they are indeed far away stations.

2 Background

In this section, we introduce the notation and terminology for spatial models and spatio-temporal traces and we describe Spatio-Temporal Reach and Escape Logic (STREL).

Definition 1 (Spatial Model). A spatial model \mathcal{S} is defined as a pair $\langle L, W \rangle$, where L is a set of nodes or locations and $W \subseteq L \times \mathbb{R} \times L$ is a nonempty relation associating each distinct pair $\ell_1, \ell_2 \in L$ with a label $w \in \mathbb{R}$ (also denoted $\ell_1 \xrightarrow{w} \ell_2$).

There are many different choices possible for the proximity relation W ; for example, W could be defined in a way that the edge-weights indicate spatial proximity, communication network connectivity etc. Given a set of locations, unless there is a user-specified W , we note that there are several graphs (and associated edge-weights) that we can use to express spatial models. We explore these possibilities in Sect. 3. For the rest of this section, we assume that W is defined using the notion of (δ, d) -connectivity graph as defined in Definition 2.

Definition 2 ((δ, d) -connectivity spatial model). Given a compact metric space M with the distance metric $d : M \times M \rightarrow \mathbb{R}^{\geq 0}$, a set of locations L that is a finite subset of M , and a fixed $\delta \in \mathbb{R}, \delta > 0$, a (δ, d) -connectivity spatial model is defined as $\langle L, W \rangle$, where $(\ell_1, w, \ell_2) \in W$ iff $d(\ell_1, \ell_2) = w$, and $w < \delta$.

Example 1. In the BSS, each bike station is a node/location in the spatial model, where locations are assumed to lie on the metric space defined by the 3D spherical manifold of the earth’s surface; each location is defined by its latitude and longitude, and the distance metric is the *Haversine distance*¹. Figure 2b shows the δ -connectivity graph of the Edinburgh BSS, with $\delta = 1$ km.

Definition 3 (Route). For a spatial model $\mathcal{S} = \langle L, W \rangle$, a route τ is an infinite sequence $\ell_0 \ell_1 \dots \ell_k \dots$ such that for any $i \geq 0$, $\ell_i \xrightarrow{w_i} \ell_{i+1}$.

For a route τ , $\tau[i]$ denotes the i^{th} node ℓ_i in τ , $\tau[i..]$ indicates the suffix route $\ell_i \ell_{i+1} \dots$, and $\tau(\ell)$ denotes $\min i \mid \tau[i] = \ell$, i.e. the first occurrence of ℓ in τ . Note that $\tau(\ell) = \infty$ if $\forall i \tau[i] \neq \ell$. We use $\mathcal{T}(\mathcal{S})$ to denote the set of routes in \mathcal{S} , and $\mathcal{T}(\mathcal{S}, \ell)$ to denote the set of routes in \mathcal{S} starting from $\ell \in L$. We can use routes to define the route distance between two locations in the spatial model as follows.

Definition 4 (Route Distance and Spatial Model Induced Distance). Given a route τ , the route distance along τ up to a location ℓ denoted $d_{\mathcal{S}}^{\tau}(\ell)$ is defined as $\sum_{i=0}^{\tau(\ell)} w_i$. The spatial model induced distance between locations ℓ_1 and ℓ_2 (denoted $d_{\mathcal{S}}(\ell_1, \ell_2)$) is defined as: $d_{\mathcal{S}}(\ell_1, \ell_2) = \min_{\tau \in \mathcal{T}(\mathcal{S}, \ell_1)} d_{\mathcal{S}}^{\tau}(\ell_2)$.

Note that by the above definition, $d_{\mathcal{S}}^{\tau}(\ell) = 0$ if $\tau[0] = \ell$ and ∞ if ℓ is not a part of the route (i.e. $\tau(\ell) = \infty$), and $d_{\mathcal{S}}(\ell_1, \ell_2) = \infty$ if there is no route from ℓ_1 to ℓ_2 .

Spatio-temporal Time-Series. A spatio-temporal trace associates each location in a spatial model with a time-series trace. Formally, a time-series trace x is a mapping from a time domain \mathbb{T} to some bounded and non-empty set known as the value domain \mathcal{V} . Given a spatial model $\mathcal{S} = \langle L, W \rangle$, a spatio-temporal trace σ is a function from $L \times \mathbb{T}$ to \mathcal{V} . We denote the time-series trace at location ℓ by $\sigma(\ell)$.

Example 2. Consider a spatio-temporal trace σ of the BSS defined such that for each location ℓ and at any given time t , $\sigma(\ell, t)$ is $(B(t), S(t))$, where $B(t)$ and $S(t)$ are respectively the number of bikes and empty slots at time t .

2.1 Spatio-temporal Reach and Escape Logic (STREL)

Syntax. STREL is a logic that was introduced in [2] as a formalism for monitoring spatially distributed cyber-physical systems. STREL extends Signal Temporal Logic [12] with two spatial operators, *reach* and *escape*, from which is possible to derive other three spatial modalities: *everywhere*, *somewhere* and *surround*. The syntax of STREL is given by:

$$\varphi ::= \text{true} \mid \mu \mid \neg \varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \text{U}_I \varphi_2 \mid \varphi_1 \mathcal{R}_D \varphi_2 \mid \mathcal{E}_D \varphi.$$

¹ Haversine Formula gives minimum distance between any two points on sphere by using their latitudes and longitudes.

Here, μ is an atomic predicate (AP) over the value domain \mathcal{V} . Negation \neg and conjunction \wedge are the standard Boolean connectives, while U_I is the temporal operator *until* with I being a non-singular interval over the time-domain \mathbb{T} . The operators \mathcal{R}_D and \mathcal{E}_D are spatial operators where D denotes an interval over the distances induced by the underlying spatial model, i.e., an interval over $\mathbb{R}^{\geq 0}$.

Semantics. A STREL formula is evaluated piecewise over each location and each time appearing in a given spatio-temporal trace. We use the notation $(\sigma, \ell) \models \varphi$ if the formula φ holds true at location ℓ for the given spatio-temporal trace σ . The interpretation of atomic predicates, Boolean operations and temporal operators follows standard semantics for Signal Temporal Logic: E.g., for a given location ℓ and a given time t , the formula $\varphi_1 U_I \varphi_2$ holds at ℓ iff there is some time t' in $t \oplus I$ where φ_2 holds, and for all times t'' in $[t, t')$, φ_1 holds. Here the \oplus operator defines the interval obtained by adding t to both interval end-points. We use standard abbreviations $\mathbf{F}_I \varphi = \text{true} U_I \varphi$ and $\mathbf{G}_I \varphi = \neg \mathbf{F}_I \neg \varphi$, for the *eventually* and *globally* operators. The reachability (\mathcal{R}_D) and escape (\mathcal{E}_D) operators are spatial operators. The formula $\varphi_1 \mathcal{R}_D \varphi_2$ holds at a location ℓ if there is a route τ starting at ℓ that reaches a location ℓ' that satisfies φ_2 , with a route distance $d_S^{\tau}(\ell')$ that lies in the interval D , and for all preceding locations, including ℓ , φ_1 holds true. The escape formula $\mathcal{E}_D \varphi$ holds at a location ℓ if there exists a location ℓ' at a route distance $d_S(\ell_1, \ell_2)$ that lies in the interval D and a route starting at ℓ and reaching ℓ' consisting of locations that satisfy φ . We define two other operators for notational convenience: The *somewhere* operator, denoted $\diamond_{[0,d]} \varphi$, is defined as $\text{true} \mathcal{R}_{[0,d]} \varphi$, and the *everywhere* operator, denoted $\boxplus_{[0,d]} \varphi$ is defined as $\neg \diamond_{[0,d]} \neg \varphi$, where d is a real positive value; their meaning is described in the next example.

Example 3. In the BSS, we use atomic predicates $S > 0$ and $B > 10$, and the formula $\mathbf{G}_{[0,3\text{hours}]} \diamond_{[0,1\text{km}]} (B > 10)$ is true if always within the next 3h, at a location ℓ , there is some location ℓ' at most 1km from ℓ where, the number of bikes available exceed 10. Similarly, the formula $\boxplus_{[0,1\text{km}]} \mathbf{G}_{[0,30\text{min}]} (S > 0)$ is true at a location ℓ if for all locations within 1km, for the next 30 mins, there is no empty slot.

3 Constructing a Spatial Model

In this section, we present four approaches to construct a spatial model, and discuss the pros and cons of each approach.

1. **(∞, d) -connectivity spatial model:** This spatial model corresponds to the (δ, d) -connectivity spatial model as presented in Definition 2, where we set $\delta = \infty$. We note that this gives us a fully connected graph, i.e. where $|W|$ is $O(|L|^2)$. We remark that our learning algorithm uses monitoring STREL formulas as a sub-routine, and from Lemma 2 in Appendix², we can see that

² Algorithms and Appendix of the paper are provided in the [arXiv version](#) due to lack of space.

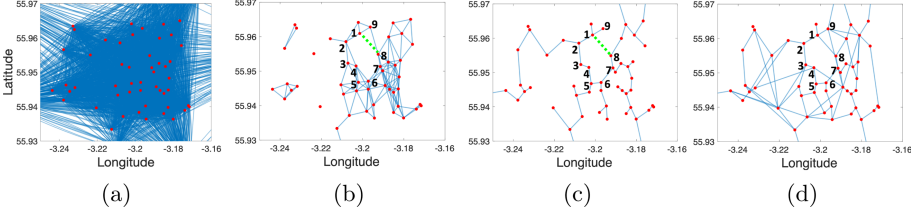


Fig. 2. Different approaches for constructing the spatial model for the BSS. (a) shows an $(\infty, d_{\text{hvrSN}})$ -connectivity spatial model where d_{hvrSN} is the Haversine distance between locations. (b) shows a $(\delta, d_{\text{hvrSN}})$ -connectivity spatial model where $\delta = 1$ km. Observe that the spatial model is disconnected. (c) shows an MST-spatial model. (d) shows an $(\alpha, d_{\text{hvrSN}})$ enhanced MSG spatial model with $\alpha = 2$. Observe that this spatial model is sparse compared even to the $(\delta, d_{\text{hvrSN}})$ -connectivity spatial model.

as the complexity of monitoring a STREL formula is linear in $|W|$, a fully connected graph is undesirable.

2. **(δ, d) -connectivity spatial model:** This is the model presented in Definition 2, where δ is heuristically chosen in an application-dependent fashion. Typically, the δ we choose is much smaller compared to the distance between the furthest nodes in the given topological space. This gives us W that is sparse, and thus with a lower monitoring cost; however, a small δ can lead to a disconnected spatial model which can affect the accuracy of the learned STREL formulas. Furthermore, this approach may overestimate the spatial model induced distance between two nodes (as in Definition 4) that are not connected by a direct edge. For instance, in Fig. 2b, nodes 1 and 8 are connected through the route $1 \rightarrow 9 \rightarrow 8$, and sum of the edge-weights along this route is larger than the actual (metric) distance of 1 and 8.
3. **MST-spatial model:** To minimize the number of edges in the graph while keeping the connectivity of the graph, we can use Minimum Spanning Tree (MST) as illustrated in Fig. 2c. This gives us $|W|$ that is $O(|L|)$, which makes monitoring much faster, while resolving the issue of disconnected nodes in the (δ, d) -spatial model. However, an MST can also lead to an overestimate of the spatial model induced distance between some nodes in the graph. For example, in Fig. 2c, the direct distance between nodes 1 and 8 is much smaller than their route distance (through the route $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$).
4. **(α, d) -Enhanced MSG Spatial Model:** To address the shortcomings of previous approaches, we propose constructing a spatial model that we call the (α, d) -Enhanced Minimum Spanning Graph Spatial model. First, we construct an MST over the given set of locations and use it to define W and pick α as some number greater than 1. Then, for each distinct pair of locations ℓ_1, ℓ_2 , we compute the shortest route distance $d_S(\ell_1, \ell_2)$ between them in the constructed MST, and compare it to their distance $d(\ell_1, \ell_2)$ in the metric space. If $d_S(\ell_1, \ell_2) > \alpha \cdot d(\ell_1, \ell_2)$, then we add an edge $(\ell_1, d(\ell_1, \ell_2), \ell_2)$ to W .

The resulting spatial model is no longer a tree, but typically is still sparse³. In our case studies, the cost of building the enhanced MSG spatial model was insignificant compared to the other steps in the learning procedure⁴.

4 Learning STREL Formulas from Data

In this section, we first introduce Parametric Spatio-Temporal Reach and Escape Logic (PSTREL) and the notion of monotonicity for PSTREL formulas. Then, we introduce a projection function π that maps a spatio-temporal trace to a valuation in the parameter space of a given PSTREL formula. We then cluster the trace-projections using Agglomerative Hierarchical Clustering, and finally learn a compact STREL formula for each cluster using Decision Tree techniques.

Parametric STREL (PSTREL). Parametric STREL (PSTREL) is a logic obtained by replacing one or more numeric constants appearing in STREL formulas by parameters; parameters appearing in atomic predicates are called *magnitude* parameters \mathcal{P}_γ , and those appearing in temporal and spatial operators are called *timing* $\mathcal{P}_\mathbb{T}$ and *spatial* parameters \mathcal{P}_{d_s} respectively. Each parameter in \mathcal{P}_γ take values from \mathcal{V} , those in $\mathcal{P}_\mathbb{T}$ take values from \mathbb{T} , and those in \mathcal{P}_{d_s} take values from $\mathbb{R}^{\geq 0}$ (i.e. the set of values that the d_s metric can take for a given spatial model). We define a valuation function ν that maps all parameters in a PSTREL formula to their respective values.

Example 4. Consider the PSTREL versions of the STREL formulas introduced in Example 3 $\varphi(\mathbf{p}_\tau, \mathbf{p}_d, \mathbf{p}_c) = \mathbf{G}_{[0, \mathbf{p}_\tau]} \diamond_{[0, \mathbf{p}_d]} (B > \mathbf{p}_c)$. The valuation $\nu: \mathbf{p}_\tau \mapsto 3\text{h}, \mathbf{p}_d \mapsto 1\text{km}, \text{ and } \mathbf{p}_c \mapsto 10$ returns the STREL formula introduced in Example 3.

Definition 5 (Parameter Polarity, Monotonic PSTREL). A *polarity function* γ maps a parameter to an element of $\{+, -\}$, and is defined as follows:

$$\begin{aligned} \gamma(\mathbf{p}) = + &\stackrel{\text{def}}{=} \nu'(\mathbf{p}) > \nu(\mathbf{p}) \wedge (\sigma, \ell) \models \varphi(\nu(\mathbf{p})) \Rightarrow (\sigma, \ell) \models \varphi(\nu'(\mathbf{p})) \\ \gamma(\mathbf{p}) = - &\stackrel{\text{def}}{=} \nu'(\mathbf{p}) < \nu(\mathbf{p}) \wedge (\sigma, \ell) \models \varphi(\nu(\mathbf{p})) \Rightarrow (\sigma, \ell) \models \varphi(\nu'(\mathbf{p})) \end{aligned}$$

The *monotonic fragment of PSTREL* consists of PSTREL formulas where all parameters have either positive or negative polarity.

³ The complete algorithm, Algorithm 1 is provided in the [arXiv version](#). Algorithm 1 is a simple way of constructing an (α, d) -enhanced MSG spatial model, and incurs a one-time cost of $O(|L|^2 \cdot (|L| + |W| \cdot \log(|L|)))$. We believe that the time complexity can be further improved using a suitable dynamic programming based approach.

⁴ The runtimes of our learning approach for different kinds of spatial models on various case studies is illustrated in Table 1 in the [arXiv version](#).

In simple terms, the polarity of a parameter \mathbf{p} is positive if it is easier to satisfy φ as we increase the value of \mathbf{p} and is negative if it is easier to satisfy φ as we decrease the value of \mathbf{p} . The notion of polarity for PSTL formulas was introduced in [1], and we extend this to PSTREL and spatial operators. The polarity for PSTREL formulas $\varphi(d_1, d_2)$ of the form $\diamond_{[d_1, d_2]}\psi$, $\psi_1 \mathcal{R}_{[d_1, d_2]}\psi_2$, and $\mathcal{E}_{[d_1, d_2]}\psi$ are $\gamma(d_1) = -$ and $\gamma(d_2) = +$, i.e. if a spatio-temporal trace satisfies $\varphi(\nu(d_1), \nu(d_2))$, then it also satisfies any STREL formula over a strictly larger spatial model induced distance interval, i.e. by decreasing $\nu(d_1)$ and increasing $\nu(d_2)$. For a formula $\square_{[d_1, d_2]}\psi$, $\gamma(d_1) = +$ and $\gamma(d_2) = -$, i.e. the formula obtained by strictly shrinking the distance interval. The proofs are simple, and provided in Appendix for completeness.

Definition 6 (Validity Domain, Boundary). *Let $P = \mathcal{V}^{|\mathcal{P}_V|} \times \mathbb{T}^{|\mathcal{P}_T|} \times (\mathbb{R}^{\geq 0})^{|\mathcal{P}_{ds}|}$ denote the space of parameter valuations, then the validity domain V of a PSTREL formula at a location ℓ with respect to a set of spatio-temporal traces Σ is defined as follows: $V(\varphi(\mathbf{p}), \ell, \Sigma) = \{\nu(\mathbf{p}) \mid \mathbf{p} \in P, \sigma \in \Sigma, (\sigma, \ell) \models \varphi(\nu(\mathbf{p}))\}$ The validity domain boundary $\partial V(\varphi(\varphi), \ell, \Sigma)$ is defined as the intersection of $V(\varphi, \ell, \Sigma)$ with the closure of its complement.*

Spatio-temporal Trace Projection. We now explain how a monotonic PSTREL formula $\varphi(\mathbf{p})$ can be used to automatically extract features from a spatio-temporal trace. The main idea is to define a total order $>_{\mathcal{P}}$ on the parameters \mathbf{p} (i.e. parameter priorities) that allows us to define a lexicographic projection of the spatio-temporal trace σ at each location ℓ to a parameter valuation $\nu(\mathbf{p})$ (this is similar to assumptions made in [8, 11]). We briefly remark how we can relax this assumption later. Let ν_j denote the valuation of the j^{th} parameter.

Definition 7 (Parameter Space Ordering, Projection). *A total order on parameter indices $j_1 > \dots > j_n$ imposes a total order $<_{\text{lex}}$ on the parameter space defined as:*

$$\nu(\mathbf{p}) <_{\text{lex}} \nu'(\mathbf{p}) \Leftrightarrow \exists j_k \text{ s.t. } \begin{cases} \gamma(\mathbf{p}_{j_k}) = + \Rightarrow \nu_{j_k} < \nu'_{j_k} \\ \gamma(\mathbf{p}_{j_k}) = - \Rightarrow \nu_{j_k} > \nu'_{j_k} \end{cases} \text{ and } \forall m <_{\mathcal{P}} k, \nu_m = \nu'_m.$$

Given above total order, $\pi_{\text{lex}}(\sigma, \ell) = \inf_{<_{\text{lex}}} \{\nu(\mathbf{p}) \in \partial V(\varphi(\mathbf{p}), \{\sigma\})\}$.

In simple terms, given a total order on the parameters, the lexicographic projection maps a spatio-temporal trace to valuations that are least permissive w.r.t. the parameter with the greatest priority, then among those valuations, to those that are least permissive w.r.t. the parameter with the next greater priority, and so on. Finding a lexicographic projection can be done by sequentially performing binary search on each parameter dimension [11]⁵. It is easy to show that π_{lex} returns a valuation on the validity domain boundary.

⁵ Algorithm 2 is provided in the [arXiv version](#).

Remark 1. The order of parameters is assumed to be provided by the user and is important as it affects the unsupervised learning algorithms for clustering that we apply next. Intuitively, the order corresponds to what the user deems as more important. For example, consider the formula $\mathbf{G}_{[0,3\text{hours}]} \diamond_{[0,d]} (B > c)$. Note that $\gamma(d) = +$, and $\gamma(c) = -$. Now if the user is more interested in the radius around each station where the number of bikes exceeds some threshold (possibly 0) within 3 h, then the order is $d >_{\mathcal{P}} c$. If she is more interested in knowing what is the largest number of bikes available in any radius (possibly ∞) always within 3 h, then $c >_{\mathcal{P}} d$.

Remark 2. Similar to [18], we can compute an approximation of the validity domain boundary for a given trace, and then apply a clustering algorithm on the validity domain boundaries. This does not require the user to specify parameter priorities. In all our case studies, the parameter priorities were clear from the domain knowledge, and hence we will investigate this extension in the future.

Clustering. The projection operator $\pi_{\text{lex}}(\sigma, \ell)$ maps each location to a valuation in the parameter space. These valuation points serve as features for off-the-shelf clustering algorithms. In our experiments, we use the *Agglomerative Hierarchical Clustering* (AHC) technique [5] to automatically cluster similar valuations. AHC is a bottom-up approach that starts by assigning each point to a single cluster, and then merging clusters in a hierarchical manner based on a similarity criteria⁶. An important hyperparameter for any clustering algorithm is the number of clusters to choose. In some case studies, we use domain knowledge to decide the number of clusters. Where such knowledge is not available, we use the *Silhouette metric* to compute the optimal number of clusters. Silhouette is a ML method to interpret and validate consistency within clusters by measuring how well each point has been clustered. The silhouette metric ranges from -1 to $+1$, where a high silhouette value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters [17].

Example 5. Figure 1a shows the results of projecting the spatio-temporal traces from BSS through the PSTREL formula $\varphi(\tau, d)$ shown in Eq. (1).

$$\varphi(\tau, d) = \mathbf{G}_{[0,3]}(\varphi_{\text{wait}}(\tau) \vee \varphi_{\text{walk}}(d)) \quad (1)$$

In the above formula, $\varphi_{\text{wait}}(\tau)$ is defined as $\mathbf{F}_{[0,\tau]}(B \geq 1) \wedge (\mathbf{F}_{[0,\tau]}S \geq 1)$, and $\varphi_{\text{walk}}(d)$ is $\diamond_{[0,d]}(B \geq 1) \wedge \diamond(S \geq 1)$. $\varphi(\tau, d)$ means that for the next 3 h, either $\varphi_{\text{wait}}(\tau)$ or $\varphi_{\text{walk}}(d)$ is true. Locations with large values of τ have long wait times or with large d values are typically far from a location with bike/slot availability (and are thus undesirable). Locations with small τ, d are desirable. Each point in Fig. 1a shows $\pi_{\text{lex}}(\sigma, \ell)$ applied to each location and the result of applying AHC with 3 clusters.

⁶ We used complete-linkage criteria which assumes the distance between clusters equals the distance between those two elements (one in each cluster) that are farthest away from each other.

Let $numC$ be the number of clusters obtained after applying AHC to the parameter valuations. Let C denote the labeling function mapping $\pi_{\text{lex}}(\sigma, \ell)$ to $\{1, \dots, numC\}$. The next step after clustering is to represent each cluster in terms of an easily interpretable STREL formula. Next, we propose a decision tree-based approach to learn an interpretable STREL formula from each cluster.

Learning STREL Formulas from Clusters. The main goal of this subsection is to obtain a compact STREL formula to describe each cluster identified by AHC. We argue that bounded length formulas tend to be human-interpretable, and show how we can automatically obtain such formulas using a decision-tree approach. Decision-trees (DTs) are a non-parametric supervised learning method used for classification and regression[13]. Given a finite set of points $X \subseteq \mathbb{R}^m$ and a labeling function \mathcal{L} that maps each point $x \in X$ to some label $\mathcal{L}(x)$, the DT learning algorithm creates a tree whose non-leaf nodes n_j are annotated with constraints ϕ_j , and each leaf node is associated with some label in the range of \mathcal{L} . Each path n_1, \dots, n_i, n_{i+1} from the root node to a leaf node corresponds to a conjunction $\bigwedge_{j=1}^i h_j$, where $h_j = \neg\phi_j$ if n_{j+1} is the left child of n_j and ϕ_j otherwise. Each label thus corresponds to the disjunction over the conjunctions corresponding to each path from the root node to the leaf node with that label.

Recall that after applying the AHC procedure, we get one valuation $\pi_{\text{lex}}(\sigma, \ell)$ for each location, and its associated cluster label. We apply a DT learning algorithm to each point $\pi_{\text{lex}}(\sigma, \ell)$, and each DT node is associated with a ϕ_j of the form $p_j \geq v_j$ for some $p_j \in \mathbf{p}$.

Lemma 1. *Any path in the DT corresponds to a STREL formula of length that is $O((|\mathcal{P}| + 1) \cdot |\varphi|)$.*

Proof. Any path in the DT is a conjunction over a number of formulas of the kind $p_j \geq v_j$ or its negation. Because $\varphi(\mathbf{p})$ is monotonic in each of its parameters, if we are given a conjunction of two conjuncts of the type $p_j \geq v_j$ and $p_j \geq v'_j$, then depending on $\gamma(p_j)$, one inequality implies the other, and we can discard the weaker inequality. Repeating this procedure, for each parameter, we will be left with at most 2 inequalities (one specifying a lower limit and the other an upper limit on p_j). Thus, each path in the DT corresponds to an axis-aligned hyperbox in the parameter space. Due to monotonicity, an axis-aligned hyperbox in the parameter space can be represented by a formula that is a conjunction of $|\mathcal{P}| + 1$ STREL formulas (negations of formulas corresponding to the $|\mathcal{P}|$ vertices connected to the vertex with the most permissive STREL formula, and the most permissive formula itself) [11] (see Fig. 3a for an example in a 2D parameter space). Thus, each path in the DT can be described by a formula of length $O((|\mathcal{P}| + 1) \cdot |\varphi|)$, where $|\varphi|$ is the length of φ .

Example 6. The result of applying the DT algorithm to the clusters identified by AHC (shown in dotted lines in Fig. 1a) is shown as the axis-aligned hyperboxes. Using the meaning of $\varphi(\tau, d)$ as defined in Eq. (1), we learn the formula $\neg\varphi(17.09, 2100) \wedge \neg\varphi(50, 1000.98) \wedge \varphi(50, 2100)$ for the red cluster. The last of these conjuncts is essentially the formula *true*, as this formula corresponds to

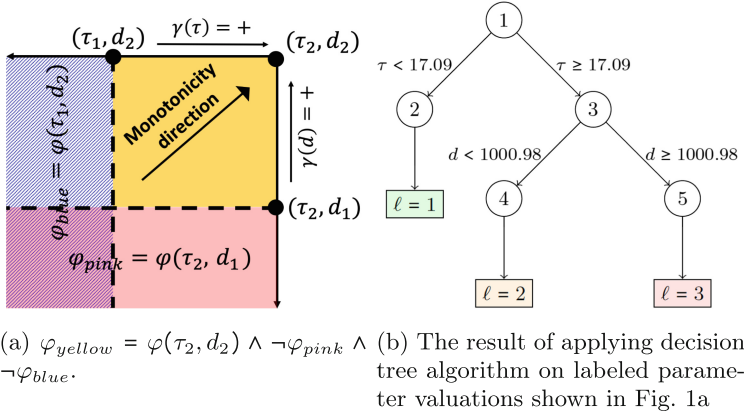


Fig. 3. Illustration of clustering on the BSS locations

the most permissive formula over the given parameter space. Thus, the formula we learn is:

$$\varphi_{red} = \neg\mathbf{G}_{[0,3]}(\varphi_{wait}(17.09) \vee \varphi_{walk}(2100)) \wedge \neg\mathbf{G}_{[0,3]}(\varphi_{wait}(50) \vee \varphi_{walk}(1000.98))$$

The first of these conjuncts is associated with a short wait time and the second is associated with short walking distance. As both are not satisfied, these locations are the least desirable.

Pruning the Decision Tree. If the decision tree algorithm produces several disjuncts for a given label (e.g., see Fig. 4a), then it can significantly increase the length and complexity of the formula that we learn for a label. This typically happens when the clusters produced by AHC are not clearly separable using axis-aligned hyperplanes. We can mitigate this by pruning the decision tree to a maximum depth, and in the process losing the bijective mapping between cluster labels and small STREL formulas. We can still recover an STREL formula that is satisfied by most points in a cluster using a k -fold cross validation approach (The formal procedure is presented in Algorithm 3 in the [arXiv version](#).) The idea is to loop over the maximum depth permitted from 1 to N , where N is user provided, and for each depth performing k -fold cross validation to characterize the accuracy of classification at that depth. If the accuracy is greater than a threshold (90% in our experiments), we stop and return the depth as a limit for the decision tree. Figure 4b illustrates the hyper-boxes obtained using this approach. For this example, we could decrease the number of hyper-boxes from 11 to 3 by miss-classifying only a few data points (less than 10% of the data).

5 Case Studies

We now present the results of applying the clustering techniques developed on three benchmarks: (1) COVID-19 data from Los Angeles County, USA, [9] (2)

Outdoor Air Quality data from California, and (3) BSS data from the city of Edinburgh [10] (running example)⁷. A summary of the computational aspects of the results is provided in Table 1. The numbers indicate that our methods scale to spatial models containing hundreds of locations, and still learn interpretable STREL formulas for clusters.

Table 1. Summary of results.

Case	$ L $	$ W $	Run-time (secs)	$numC$	$ \varphi_{cluster} $
COVID-19	235	427	813.65	3	$3 \cdot \varphi + 4$
BSS	61	91	681.78	3	$2 \cdot \varphi + 4$
Air Quality	107	60	136.02	8	$5 \cdot \varphi + 7$
Food Court*	20	35	78.24	8	$3 \cdot \varphi + 4$

COVID-19 Data from LA County. Understanding the spread pattern of COVID-19 in different areas is vital to stop the spread of the disease. While this example is not related to a software system, it is nevertheless a useful example to show the versatility of our approach to spatio-temporal data. The PSTREL formula $\varphi(c, d) = \diamond_{[0, d]}(\mathbf{F}_{[0, \tau]}(x > c))$ allows us to number of cases exceeding a threshold c within $\tau = 10$ days in a neighborhood of size d for a given location⁸. Locations with small value of d and large value of c are unsafe as there is a large number of new positive cases within a small radius around them.

We illustrate the clustering results in Fig. 4. Each location in Fig. 4a is associated with a geographic region in LA county (shown in Fig. 4c), and the *red* cluster corresponds to hot spots (small d and large c). Applying the DT classifier on the learned clusters (shown in Fig. 4a) produces 11 hyperboxes, some of which contain only a few points. Hence we apply our DT pruning procedure to obtain the largest cluster that gives us at least 90% accuracy. Figure 4b shows the results after pruning the Decision Tree. We learn the following formula:

$$\varphi_{red} = \diamond_{[0, 4691.29]}(\mathbf{F}_{[0, 10]}(x > 3180)) \vee \diamond_{[0, 15000]}(\mathbf{F}_{[0, 10]}(x > 5611.5)),$$

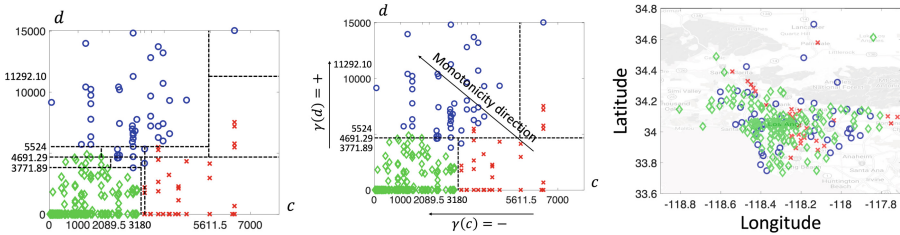
⁷ We provide results on a fourth benchmark consisting of a synthetic dataset for tracking movements of people in a food court building and detailed descriptions for each benchmark in the Appendix. All experiments were performed on an Intel Core-i7 Macbook Pro with 2.7 GHz processor and 16 GB RAM. We use an existing monitoring tool MoonLight [3] in Matlab for computing the robustness of STREL formulas. For Agglomerative Hierarchical Clustering and Decision Tree techniques we use scikit-learn library in Python and the Statistics and Machine Learning Toolbox in Matlab.

⁸ We fix τ to 10 days and focus on learning the values of c and d for each location.

This formula means that within 4691.29m from any *red* location, within 10 days, the number of new positive cases exceeds 3180. The COVID-19 data that we used is for September 2020⁹.

Outdoor Air Quality Data from California. We next consider Air Quality data from California gathered by the US Environmental Protection Agency (EPA). Among reported pollutants we focus on *PM2.5* contaminant, and try to learn the patterns in the amount of *PM2.5* in the air using STREL formulas. Consider a mobile sensing network consisting of UAVs to monitor pollution, such a STREL formula could be used to characterize locations that need increased monitoring.

We use the PSTREL formula $\varphi(c, d) = \mathbf{G}_{[0,10]}(\mathcal{E}_{[d,16000]}(PM2.5 < c))$ and project each location in California to the parameter space of c, d . A location ℓ satisfies this property if it is always true within the next 10 days, that there exists a location ℓ' at a distance more than d , and a route τ starting from ℓ and reaching ℓ' such that all the locations in the route satisfy the property $PM2.5 < c$. Hence, it might be possible to escape to a location at a distance greater than d always satisfying property $PM2.5 < c$. The results are shown in Fig. 5a. Cluster 8 is the best cluster as it has a small value of c and large value of d which means that there exists a long route from the locations in cluster 8 with low density of *PM2.5*. Cluster 3 is the worst as it has a large value of c and a small value of d . The formula for cluster 3 is $\varphi_3 = \varphi(500, 0) \wedge \neg\varphi(500, 2500) \wedge \neg\varphi(216, 0)$. φ_3 holds in locations where, in the next 10 days, *PM2.5* is always less than 500, but at least in 1 day *PM2.5* reaches 216 and there is no safe route (i.e. locations along the route have $PM2.5 < 500$) of length at least 2500.



(a) The learned hyper-boxes before pruning the DT. (b) The learned hyper-boxes after pruning the DT. (c) Red-color points: hot spots. Green-color points: spots.

Fig. 4. Procedure to learn STREL formulas from COVID-19 data

⁹ In Fig. 6 in the appendix of the [arXiv version](#), we show the results of STREL clustering for 3 different months in 2020, which confirms the rapid spread of the COVID-19 virus in LA county from April 2020 to September 2020. Furthermore, we can clearly see spread of the virus around the hot spots during the time, a further validation of our approach.

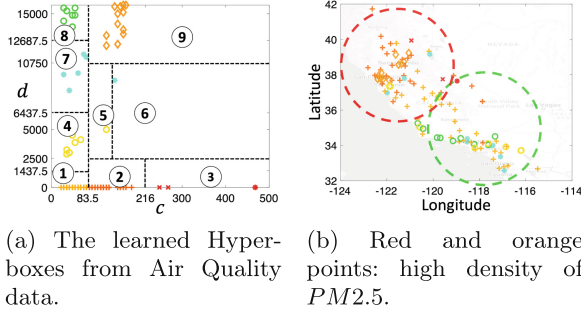


Fig. 5. Clustering experiments on the California Air Quality Data

6 Related Work and Conclusion

Traditional ML Approaches for Time-Series Clustering. Time-series clustering is a popular area in the domain of machine learning and data mining. Some techniques for time-series clustering combine clustering methods such as KMeans [7], Hierarchical Clustering, agglomerative clustering [4] and etc., with similarity metrics between time-series data such as the Euclidean distance, dynamic time-warping (DTW) distance, and statistical measures (such as mean, median, correlation, etc.). Some recent works such as the works on shapelets automatically identify distinguishing shapes in the time-series data [19]. Such shapelets serve as features for ML tasks. All these approaches are based on shape-similarity which might be useful in some applications; however, for applications that the user is interested in mining temporal information from data, dissimilar traces might be clustered in the same group [11]. Furthermore, such approaches may lack interpretability as we showed in BSS case study.

STL-Based Clustering of Time-Series Data. There is considerable amount of recent work on learning temporal logic formulas from time-series data using logics such as Signal Temporal Logic (STL) [8, 11, 14, 15]; however, there is no work on discovering such relations on spatio-temporal data. In particular, the work in [11] which addresses unsupervised clustering of time-series data using Signal Temporal Logic is closest to our work. There are a few hurdles in applying such an approach to spatio-temporal data as explained in Sect. 1. We address all the hurdles in the current work.

Monitoring Spatio-temporal Properties. There is considerable amount of recent work such as [2, 3] on monitoring spatio-temporal properties. Particularly, MoonLight [3] is a recent tool for monitoring of STREL properties, and in our current work, we use MoonLight for computing the robustness of spatio-temporal data with respect to STREL formulas. MoonLight uses (δ, d) -connectivity approach for creating a spatial model, which has several issues, including disconnectivity and distance overestimation. We resolve these issues by proposing our new method for creating the spatial graph, which we call Enhanced MSG.

While there are many works on monitoring of spatio-temporal logic, to the best of our knowledge, there is no work on automatically inferring spatio-temporal logic formulas from data that we address in this work.

Conclusion. In this work, we proposed a technique to learn interpretable STREL formulas from spatio-temporal time-series data for Spatially Distributed Systems. First, we introduced the notion of monotonicity for a PSTREL formula, proving the monotonicity of each spatial operator. We proposed a new method for creating a spatial model with a restrict number of edges that preserves connectivity of the spatial model. We leveraged quantitative semantics of STREL combined with multi-dimensional bisection search to extract features for spatio-temporal time-series clustering. We applied Agglomerative Hierarchical clustering on the extracted features followed by a Decision Tree based approach to learn an interpretable STREL formula for each cluster. We then illustrated with a number of benchmarks how this technique could be used and the kinds of insights it can develop. The results show that while our method performs slower than traditional ML approaches, it is more interpretable and provides a better insight into the data. For future work, we will study extensions of this approach to supervised and active learning.

Acknowledgments. We thank the anonymous reviewers for their comments. The authors also gratefully acknowledge the support by the National Science Foundation under the Career Award SHF-2048094, the NSF FMitF award CCF-1837131, the Austrian FWF projects ZK-35, and a grant from Toyota R&D North America.

References

1. Asarin, E., Donzé, A., Maler, O., Nickovic, D.: Parametric identification of temporal properties. In: Khurshid, S., Sen, K. (eds.) RV 2011. LNCS, vol. 7186, pp. 147–160. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29860-8_12
2. Bartocci, E., Bortolussi, L., Loretì, M., Nenzi, L.: Monitoring mobile and spatially distributed cyber-physical systems. In: Proceedings of MEMOCODE (2017)
3. Bartocci, E., Bortolussi, L., Loretì, M., Nenzi, L., Silvetti, S.: MoonLight: a lightweight tool for monitoring spatio-temporal properties. In: Deshmukh, J., Ničković, D. (eds.) RV 2020. LNCS, vol. 12399, pp. 417–428. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60508-7_23
4. Cobo, G., García-Solórzano, D., Santamaría, E., Morán, J.A., Melenchón, J., Monzo, C.: Modeling students' activity in online discussion forums: a strategy based on time series and agglomerative hierarchical clustering. In: Educational Data Mining (2010)
5. Day, W.H., Edelsbrunner, H.: Efficient algorithms for agglomerative hierarchical clustering methods. *J. Classif.* **1**(1), 7–24 (1984)
6. Fiedler, B., Scheel, A.: Spatio-temporal dynamics of reaction-diffusion patterns. In: Kirkilionis, M., Krömker, S., Rannacher, R., Tomi, F. (eds.) Trends in Nonlinear Analysis, pp. 23–152. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-662-05281-5_2

7. Huang, X., Ye, Y., Xiong, L., Lau, R.Y., Jiang, N., Wang, S.: Time series k-means: a new k-means type smooth subspace clustering for time series data. *Inf. Sci.* **367**, 1–13 (2016)
8. Jin, X., Donzé, A., Deshmukh, J.V., Seshia, S.A.: Mining requirements from closed-loop control models. *IEEE Trans. CAD* **34**(11), 1704–1717 (2015)
9. Kiamari, M., Ramachandran, G., Nguyen, Q., Pereira, E., Holm, J., Krishnamachari, B.: Covid-19 risk estimation using a time-varying sir-model. In: Proceedings of the 1st ACM SIGSPATIAL International Workshop on Modeling and Understanding the Spread of COVID-19, pp. 36–42 (2020)
10. Kreikemeyer, J.N., Hillston, J., Uhrmacher, A.: Probing the performance of the Edinburgh bike sharing system using SSTL. In: Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, pp. 141–152 (2020)
11. Vazquez-Chanlatte, M., Deshmukh, J.V., Jin, X., Seshia, S.A.: Logical clustering and learning for time-series data. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 305–325. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_15
12. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Lakhnech, Y., Yovine, S. (eds.) FORMATS/FTRTFT -2004. LNCS, vol. 3253, pp. 152–166. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30206-3_12
13. Mitchell, T.M.: *Machine Learning*, 1st edn. McGraw-Hill Inc., New York (1997)
14. Mohammadinejad, S., Deshmukh, J.V., Puranic, A.G.: Mining environment assumptions for cyber-physical system models. In: Proceedings of ICCPS (2020)
15. Mohammadinejad, S., Deshmukh, J.V., Puranic, A.G., Vazquez-Chanlatte, M., Donzé, A.: Interpretable classification of time-series data using efficient enumerative techniques. In: Proceedings of HSCC (2020)
16. Nenzi, L., Bortolussi, L., Ciancia, V., Loreti, M., Massink, M.: Qualitative and quantitative monitoring of spatio-temporal properties with SSTL. *LMCS* **14**(4) (2018)
17. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **20**, 53–65 (1987)
18. Vazquez-Chanlatte, M., Ghosh, S., Deshmukh, J.V., Sangiovanni-Vincentelli, A., Seshia, S.A.: Time-series learning using monotonic logical properties. In: Colombo, C., Leucker, M. (eds.) RV 2018. LNCS, vol. 11237, pp. 389–405. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03769-7_22
19. Zakaria, J., Mueen, A., Keogh, E.: Clustering time series using unsupervised-shapelets. In: 2012 IEEE 12th International Conference on Data Mining, pp. 785–794. IEEE (2012)